FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) - Barcelona Tech

Design of an IOT open platform

Autor: Aleix Sanfeliu
Directora: Cristina Gómez Seoane
Departament d'Enginyeria de Serveis i Sistemes d'Informació

Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona
Grau en Enginyeria Informàtica
Enginyeria del Software

25 de Abril 2019

Resum

En aquest projecte s'ha redissenyat i afegit noves funcionalitats al backoffice de la StartUp GreenCustomers. La nova arquitectura i aplicació, les quals formen part d'un sistema IoT, permetran obtenir una plataforma oberta, escalable i versàtil capaç d'adaptar-se a futures circumstàncies i integrar-se amb noves fonts d'informació i amb altres components de l'empresa a través d'APIs.

Resumen

En este proyecto se ha desarrollado y añadido nuevas funcionalidades al backoffice de la StartUp GreenCustomers. La nueva arquitectura y aplicación, las cuales forman parte de un sistema IoT, permitirán obtener una plataforma abierta, escalable y versátil capaz de adaptarse a futuras circunstancias e integrarse con nuevas fuentes de información y otros components de la empresa usando APIs.

Abstract

In this project the backoffice from the StartUp GreenCustomers has been redesigned and also new features have been added. The new architecture and application, which are part of an IoT system, will turn the platform to an open, scalable and versatile one able of adapt itself to future circumstances and integrate with new information sources as well as with other internal components through APIs.

Glossari

IoT	(o Internet de les coses en les sigles en anglès), és una xarxa de dispositius físics capaços de connectar-se entre si i poder recollir i intercanviar informació.
Web App	(o aplicacions web), és un programa en què interactua un client amb un servidor, i que s'executa sobre un explorador web.
CEO	(o cap executiu), és l'executiu o administrador principal d'una organització o empresa.
CIO	(o cap d'innovació), és la persona encarregada en una empresa responsable dels processos d'innovació i millora.
BackOffice	és un software que té únicament interacció amb els membres interns d'una empresa.
FrontOffice	és un software orientat a la interacció amb els clients d'una empresa.
BackEnd	és la capa d'accés a les dades d'una aplicació. En termes d'interacció client-servidor, es considera el servidor el *backend* de l'aplicació.
FrontEnd	és la capa de presentació d'una aplicació. En termes d'interacció client-servidor, es considera el client el *frontend* de l'aplicació.

Índex

Índex de taules i figures

1 Introducció

1.1 Context

Aquest projecte s'emmarca en la Normativa del Treball de Fi de Grau aprovada el 21/10/2015. La matriculació d'aquest és en la modalitat A: treball que es realitza a la UPC. Tot i la modalitat triada, aquest projecte es realitzarà conjuntament amb la StartUp GreenCustomers SL.

GreenCustomers és una StartUp que neix el 2016 per a donar solució a un dels problemes mediambientals a què els hotels s'enfronten diàriament: l'impacte ecològic dels consums d'aigua i energia que es generen en una estada. Els seus objectius de mercat no són només els hotelers sinó que també ho són els hostes. Als hotelers se'ls proporciona tota la informació relativa al consum de l'hotel i quina és la petjada ecològica generada, així com el coneixement de com poder reduir-la. I als segons se'ls permet saber quin ha estat el balanç energètic de la seva estada.

Actualment l'única via existent amb què els hotelers són capaços de comprendre l'impacte ecològic és mitjançant auditories. Els auditors però, realitzen l'auditoria en l'àmbit general de l'hotel i els és impossible saber el consum real que es genera habitació per habitació. GreenCustomers busca obtenir aquest coneixement. Per dur a terme aquesta tasca, l'empresa va crear la *greenroom*. La *greenroom* és el distintiu d'una habitació 'verda' que indica que l'habitació està connectada i permet mesurar el compromís de l'hotel i de l'hoste amb el medi ambient.

De forma molt resumida, la finalitat de l'empresa és aportar als hotels les eines i el coneixement per a obtenir informació més detallada i poder desenvolupar plans d'estalvi i millora en l'eficiència energètica més precisos. I conscienciar als hostes de les conseqüències de les seves accions.

1.2 Formulació del problema

L'arquitectura i les aplicacions de GreenCustomers han tingut un procés de desenvolupament continu, en el qual, durant aquests quasi tres anys, s'han construït de forma incremental a mesura que sorgien noves necessitats per part de l'empresa o requisits per part dels clients.

La primera eina que es va desenvolupar va ser el portal pels hotelers amb funcionalitats molt bàsiques. Però a mesura que aquests tenien nous requeriments, es van afegir funcionalitats en l'aplicació. Aquest creixement de la lògica i complexitat va induir en la creació d'un *backoffice* per a poder prendre el control dels components del sistema. Les exigències dels hotelers en els terminis i els seus canvis d'opinió, deguts al desconeixement del que realment buscaven i de ser el primer cop que havien de plantejar-se aquestes qüestions, van propiciar buscar solucions ràpides i de baix cost a curt termini.

De forma progressiva, la interdependència entre el portal i el backoffice, incrementava la dificultat d'inserció de canvis i/o noves característiques. El que hauria de ser una infraestructura amb serveis clarament definits i independents, ha evolucionat a un seguit de sistemes fortament lligats fet que escapa de la definició d'un bon software: el qual ha de ser fàcilment canviable i mantenible.

La web app per als hostes es va crear un cop ja es tenia constància de la problemàtica entre el portal i el backoffice. Per evitar que l'aplicació dels hostes estigués fortament lligada, es var decidir executar-la en un entorn amb la seva pròpia base de dades. Es va preferir crear una duplicitat en les dades de consum, davant del risc d'afegir un tercer element en les dependències entre el portal i el backoffice.

Aquest projecte, per tant, es presenta per a resoldre la problemàtica en les dependències entre el backoffice, el portal i les bases de dades redefinint i simplificant la infraestructura per tal de poder allargar-ne la vida útil i complir els estàndards de mantenibilitat i canviabilitat. Es tracta de crear un nou sistema com a base on poder créixer i que recollir l'experiència acumulada.

1.3 Objectius del projecte

El projecte se centra, per tant, en redissenyar el backoffice de GreenCustomers per a allargar-ne la vida útil. Més concretament els objectius que ha de complir el backoffice han estat separats en els següents tres blocs segons les seves característiques.

1.3.1 Redisseny del backoffice

L'actual backoffice disposa de panells de control, que s'han de mantenir millorant-ne la usabilitat, fiabilitat i eficiència, amb les següents funcionalitats on poder administrar els ítems amb rol dins del sistema:

- Gestió de nodes. Un node en el sistema és la representació virtual d'una antena: El component físic instal·lat en una zona emissor de les dades generades. El backoffice ha de poder donar d'alta nous nodes en el sistema, modificar-ne els existents i poder desactivar-los.

- Gestió de sensors. Un sensor en el sistema representa un sensor real connectat a un node. El sensor és l'encarregat de generar les dades de consum. Per tant, s'ha de poder donar d'alta sensors en el sistema, modificar-ne els existents i poder desactivar-los.

- Gestió de zones. Una zona és la representació d'un espai físic: pot ser tant una habitació, com qualsevol altra àrea que es vulgui monitoritzar: com pot ser la piscina de l'hotel, la cuina o la recepció. S'ha de poder donar d'alta zones en el sistema, modificar-ne les existents i desactivar-les.

- Gestió de tipus d'àrea. Un tipus d'àrea permet distingir entre àrees i defineix els valors de consums màxims i d'eficiència per a les àrees associades. S'ha de poder crear-ne, editar-ne i eliminar-ne.

- Gestió d'*energy rates*. Un *energy rate* representa un tipus tasa d'energia: com s'ha de calcular el preu de l'electricitat per a cada hotel. S'ha de poder crear-ne, editar-ne i eliminar-ne.

En el redisseny incorporarà dos panells que actualment es troben al portal:

- Gestió d'usuaris. Un usuari és la representació d'un membre de l'empresa amb accés tant al portal com al backoffice. S'ha de poder crear, editar i desactivar usuaris.

- Gestió de clients. Un client és la representació d'un hoteler amb accés al portal. S'ha de poder crear, editar i desactivar clients.

El nou backoffice, a més, ha d'incloure nous panells per a l'administració de nous ítems:

- Gestió de tipus de node. No tots els nodes són iguals, depenent de les necessitats de cada cas, es triarà el més adequat. Per tant, el backoffice ha de poder donar d'alta tipus de nodes en el sistema, modificar-ne els existents i poder desactivar-los.

- Gestió de components. Un component és la representació entre l'associació d'un node i una zona. S'han de poder associar nodes a una zona, reassignar-los, o desfer l'assignació.

- Gestió de zones. A més de les operacions existents, el nou backoffice ha de permetre jerarquitzar-les: que una zona formi part, amb d'altres, d'una zona superior: per exemple, que diferents zones que representen habitacions, formin part d'una altra que representa una planta d'un hotel.

1.3.2 Redisseny de l'end-point IoT

Dotar el nou backoffice de la capacitat de rebre les dades generades a les greenrooms, normalitzar-les i emmagatzemar-les de forma coherent a les mètriques i estadístiques que se'n voldran extreure.

- GreenCustomers es basa en la captació de dades, principalment, de les habitacions d'hotel. Per tant, el nou sistema, ha de mantenir la capacitat de rebre i tractar correctament, almenys, les dades provinents de la font usada actualment. Això implica reconèixer quin node les ha originat i extreure'n informació amb valor.

- Addicionalment, i amb previsió, preparar el nou sistema perquè sigui capaç de rebre dades de noves fonts i poder-les normalitzar de forma correcta independentment de l'origen.

- Emmagatzemar la informació normalitzada i assignar-la a la zona en la qual s'ha originat.

- Crear un registre per a cada trama de dades rebuda i etiquetar-la com a correcte o activar una alarma en cas d'error. Les alarmes són events definits que detecten errors en el procés IoT: des de la captació, el processat i emmagatzematge dels consums generats.

- Poder consultar el registre de trames i filtrar-ne per node, rangs de dates, tipus de node i si han estat correctament guardades o han activat alguna alarma.

1.3.3 Creació d'una API

Crear una API perquè els clients, a través del portal web, puguin consultar l'estat del seu hotel i conèixer les mètriques extretes.

- Preparar una API on s'integrarà el portal per a l'extracció de mètriques , historials i estadístiques. Entre les opcions principals hi ha:

    - Obtenir els consums d'una zona de forma cronològica entre dues dates i agrupats en blocs de 30 minuts.

    - Obtenir els consums acumulats entre dues dates.

    - Obtenir els valors màxims, mínims i la mitjana dels consums entre dues dates i la comparació amb els del darrer any.

1.4 Actors Implicats

En el plantejament i desenvolupament del projecte hi ha un seguit de persones que tenen un especial interès en què aquest finalitzi amb èxit. Principalment, aquestes persones són els propis membres de Green Customers, per l'enfocament del projecte amb l'empresa.

Per tant, els actors implicats en aquest projecte són els següents:

- El CEO de GreenCustomers. És una de les persones més interessades en aquest projecte. És la persona encarregada d'assegurar un bon futur a l'empresa. D'aquesta forma, poder tenir a la disposició una eina robusta però alhora flexible aporta seguretat i crea una base per a seguir creixent.

- El CIO de GreenCustomers. És la persona encarregada de buscar nous proveïdors de sensors i antenes amb els quals poder reduir costos i explorar noves possibilitats. L'interès principal en el projecte se centra poder tenir una plataforma que pugui treballar amb més d'una tecnologia de comunicació. La plataforma actual està només preparada per a treballar amb Sigfox. Les principals dues noves vies de comunicació amb les que es podria operar són LoRa i WiFi.

- El desenvolupador. És la persona encarregada de desenvolupar el projecte d'inici a fi. Aquest té un interès afegit, ja que el marc en què s'encara aquest projecte és el Treball de Final de Grau (o TFG) del desenvolupador. L'èxit del projecte permetrà l'acabament del Grau satisfactòriament.

1.5 Possibles Obstacles

Durant el transcurs del projecte, és possible que apareguin factors que afectin la planificació temporal del projecte amb endarreriments. Aquests obstacles poden provenir de tres orígens distints.

Els primers són els obstacles temporals. Principalment degut als terminis marcats pel calendari. Sempre existeix la dificultat de poder acotar temporalment un projecte amb precisió. Els imprevistos poden afectar, ràpidament i greument, als terminis marcats.

El segon origen, són els problemes relacionats amb la tecnologia i/o els desconeixements sobre aquesta. Per a poder minimitzar-ne els efectes d'aquests tipus, es recorrerà a tecnologies conegudes i prèviament utilitzades en altres projectes o entorns.

El tercer i més perillós, és la pròpia planificació i disseny del projecte. Un error en una d'aquestes dues etapes pot esdevenir el principal motiu d'endarreriments en el transcurs del global del projecte.

1.6 Estat de l'art

Actualment, els sistemes IoT són presents en molts àmbits del nostre dia a dia. I més a Barcelona, una ciutat que aposta per a convertir-se en una referència mundial d'SmartCity utilitzant tecnologies IoT i el 5G.

En aquest apartat es donaran a conèixer quines són les tecnologies i empreses amb el que es podria aconseguir una solució similar. Cal tenir en compte de no confondre els sistemes domòtics o de seguiment industrial. Tot i que el procés d'obtenció de dades presenta força similituds, l'objectiu és el que es diferencia.

Per al que fa a la tecnologia, dues de les xarxes més usades en IoT, a part del 4G i el 5G, són SigFox i LoRa (o LoRaWAN):

- Sigfox és una empresa francesa que ofereix una xarxa i un protocol per permetre compartir dades de qualsevol 'objecte' des de qualsevol part del món. L'empresa s'encarrega de garantir una cobertura global de la seva xarxa i ofereix als seus usuaris (particulars i/o altres empreses) la capacitat de connectar antenes. Les dades s'envien als receptors de Sigfox que s'encarreguen de transmetre la informació al servidor i activar un callback via HTTP al servei de l'usuari [1].

- LoRa és un protocol de comunicació obert i sense ànim de lucre desenvolupat per l'empresa LoRa Alliance [2]. Aquesta plataforma, a diferència de la xarxa global de Sigfox, depèn de les xarxes locals: Els usuaris poden comunicar-se creant xarxes obertes a tothom qui hi vulgui col·laborar o bé crear-ne una de privada on només hi tindran accés les antenes pròpies del client [3]. Totes les xarxes LoRa es componen de Gateways, que actuen de receptors, i es comuniquen entre si per internet [3]. Les antenes es comuniquen amb aquests gateways directament. Això vol dir, que la recepció de les dades la realitzen els Gatways que captin les dades. Aquesta comunicació directa, per tant, pot causar la duplicitat de la trama enviada.

En aquest context és normal trobar empreses que ofereixin plataformes per al control i monitorització de dades captades per a sensors amb comptabilitat amb les tecnologies esmentades.

Aquestes empreses ofereixen plataformes genèriques per adaptar-les al nombre més gran possible de clients. De totes elles, a GreenCustomers, vam valorar-ne tres:

thethings.io és una plataforma oberta que permet a empreses implementar solucions IoT escalables i flexibles amb hardware i connectivitat agnòstica. Permet usar diferents tecnologies. En el cas de GreenCustomers: Sigfox

El seu target però són empreses de producció industrial. El seu producte se centre en tres punts del procés industrial, les quals totes treballen en real-time-data dashboards:

- *Cold Chain Tracking*: pel seguiment de la cadena de fred: Producció, transport i emmagatzematge de mercaderies que requereixen unes condiciones tèrmiques especials.

- *Location Tracking*: per al control de la localització dels vehicles de transport

- *Connected PLC* (*Power Line Communication*): per al control de la maquinària en la cadena de producció: temps operant, rendiment i estat de la màquina.

The Things Network és una xarxa IoT que opera amb la tecnologia LoRa. Aquesta comunitat et permet operar amb les xarxes públiques que altres persones i/o empreses han creat, o t'ajuda en la construcció de la teva pròpia xarxa privada.

De forma simplificada, es presenta com a una eina per a la fàcil posada en marxa de xarxes LoRa.

Per tant, tenint en compte que aquestes són només uns exemples en l'ampli mercat per al qual afecta sistemes IoT, s'ha de buscar destacar sobre aquestes solucionant un dels punts clau: el com es presenta la informació.

En tot sistema industrial és vital conèixer fins a l'últim detall l'estat de la línia de producció. I en conseqüència, es requereix un nivell tècnic mínim per a entendre el que es presenta. Des de GreenCustomers, on els clients són hotels i gestors, els quals, normalment no tenen aquest nivell, es vol apostar per la senzillesa i claredat per sobre dels tecnicismes.

1.7 Estructura de la memòria

Després de conèixer amb una pinzellada el tema sobre que tracta aquest treball, s'explicarà a fons com s'ha dut a terme el projecte. Per a tenir una idea amb més profunditat respecte de l'índex, a continuació hi ha un esquema general dels capítols i un detall sobre el contingut:

2. GreenCustomers: S'explica l'evolució de l'empresa des del seu naixement fins al dia d'avui i també quin és l'estat de l'arquitectura del sistema i quins són els seus components.

3. Anàlisi del software: S'analitzen tant els problemes i debilitats del sistema actual.

4. Definició dels requisits: Es defineixen els requisits que haurà de complir la nova arquitectura un cop el projecte estigui llest.

5. Anàlisi de solucions: Tenint en compte els requisits requerits, s'analitzen les possibles solucions als problemes i debilitats detectades, i es tria les que s'adapten millor per aconseguir el compliment dels requisits.

6. Metodologia: Es defineix quina metodologia es seguirà durant el procés de desenvolupament del projecte

7. Planificació i Gestió: Es detalla la planificació temporal i la gestió dels recursos que es durà a terme per a garantir l'èxit del projecte.

8. Redisseny de l'arquitectura: S'explica els canvis realitzats durant la transformació a la nova arquitectura.

9. Redisseny d'interfície: Es mostra com s'ha organitzat la nova interfície de l'aplicació.

10. Noves funcionalitats: Es llisten les funcionalitats afegides al sistema actual. Des de l'especificació al disseny i implementació.

11. Pla de proves: S'expliquen les proves dutes a terme per a validar la completesa dels objectius del projecte.

12. Guia d'instal·lació del software: Una guia per a instal·lar l'aplicació amb totes les seves dependències partint d'una màquina nova.

13. Resultats: Es mostren els resultats de la gestió dels recursos i s'explora la sostenibilitat del projecte.

14. Justificació del projecte: S'explica quin ha estat el global del projecte i quins coneixements i competències s'han fet servir per al seu desenvolupament.

15. Conclusions: Es mostra una visió futura del projecte i una valoració personal sobre aquest i del seu procés de desenvolupament.

2 GreenCustomers

Per a poder comprendre amb més detall el contingut del projecte és necessari definir què és GreenCustomers i com funciona de forma tecnològica.

2.1 L'empresa

GreenCustomers és una StartUp que neix el 2016 per a donar solució a un dels problemes mediambientals a què els hotels s'enfronten diàriament que no s'afronta de forma directa: l'impacte ecològic dels consums d'aigua i energia que es generen en una estada.

Per a dur a terme aquesta idea el primer que es va requerir va ser el desplegament d'un prototip funcional. Durant els primers mesos, l'objectiu va ser, per tant, obtenir un prototip amb què es poguessin recollir dades per a preparar i així poder presentar un primer producte mínim viable (o MVP). En aquest període de temps es va crear un sistema base amb el qual es fos capaç d'obtenir els resultats buscats: recollir dades de sensors d'aigua i llum, normalitzar les dades segons les especificacions tècniques, i extreure'n un històric amb les equivalències en CO2 i diners dels consums registrats.

Un cop obtingut el que es necessitava, es van realitzar les primeres proves pilot en un hotel de Barcelona. En aquesta nova etapa el portal va créixer per respondre als nous requisits i funcionalitats. Amb el pilot funcionant correctament i amb la instal·lació de greenrooms en un altre hotel, l'empresa va decidir crear el *backoffice*. El *backoffice* va servir per a controlar el sistema i les *greenrooms* i tenir constància dels sensors i antenes actius.

Gràcies al pilot, la solució es va implementar en altres hotels durant els següents mesos, i es va créixer en el nombre de *greenrooms* operatives. Paral·lelament, es van dissenyar noves eines dins la plataforma i plans futurs de creixement.

2.2 Arquitectura

L'arquitectura del sistema actual es compon de dos servidors propis amb els serveis de GreenCustomers i un servei cloud. L'arquitectura es defineix segons l'esquema:

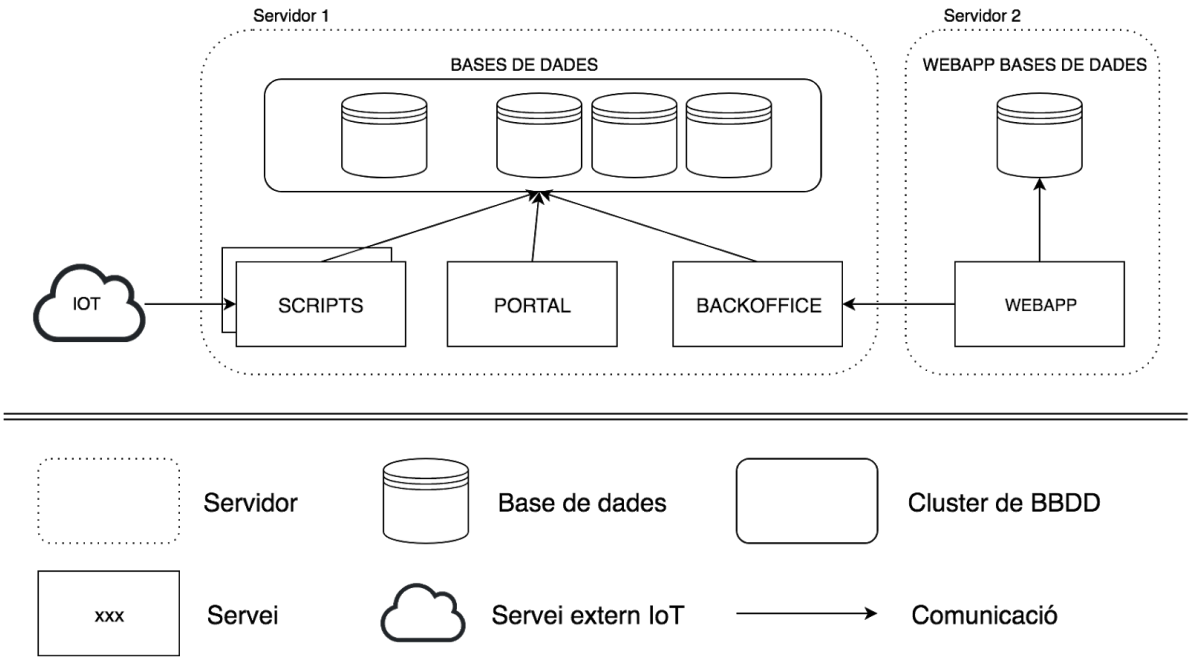


Figura 1. Estructura del sistema actual

El servei extern IoT s'utilitza com a transmissor de les dades generades als sensors de les habitacions. Aquests sensors, a través de les seves antenes, envien les dades al servei. I aquest, utilitzant un callback, les transmet al sistema.

El primer servidor conté els serveis que ens interessen més en el projecte. Són tres serveis i el clúster de bases de dades.

- El clúster de bases de dades conté una base de dades principal amb informació genèrica com els hotels, sensors i antenes actius. A més té una base de dades per hotel on s'hi emmagatzema, a part d'informació relativa l'hotel com el registre d'habitacions i alarmes generades, les dades un cop ja processades i normalitzades.

- Els scripts (de normalització) componen el servei encarregat de processar i normalitzar les dades que arriben del sistema IoT. Aquests consisteixen d'un script en python que s'encarrega de determinar la procedència de la dada i sobre quina base de dades s'ha d'afegir la nova entrada, i d'un seguit de triggers i funcions SQL en la base de dades per a calcular el valor normalitzat i actualitzar les taules corresponents.

- El backoffice, una aplicació amb *FrontEnd* i *BackEnd*, s'utilitza per controlar de forma global l'estat de GreenCustomers. Permet a l'equip gestionar els hotels i les greenrooms operatives. Això inclou els sensors i antenes muntades en les habitacions, el registre de les dades generades pre i post-normalització i si han generat alguna alarma. També els usuaris actius de la plataforma i el log de l'activitat que duen a terme.

- El portal és l'aplicació web, també amb un *FrontEnd* i un *BackEnd*, on els hotelers tenen accés per a consultar la situació de l'hotel. Això permet saber quins han estat els consums en cada habitació i en el global de les habitacions des de la seva posada en marxa. A més de l'obtenció de mètriques i estadístiques. També permet als gestors de l'hotel actualitzar la configuració de la tarifa elèctrica i d'aigua per al càlcul de l'impacte econòmic que han causat els hostes en les seves estades.

El segon servidor conté una altra aplicació, anomenada internament WebApp. Aquesta és l'aplicació web orientada als hostes. Consta d'un *FrontEnd* i un *BackEnd* propis i també de la seva base de dades on es guarden els registres de les estades de cada hoste.

2.2.1 Resum de funcionament

Un cop aclarida l'estructura del sistema, es pot explicar el funcionament bàsic de GreenCustomers: com es transmeten les dades captades dels sensors fins que l'hoteler les té accessibles.

Inicialment s'ha de fer la recaptació de les dades. Aquest es realitza a través dels sensors instal·lats a les habitacions dels hotels. Principalment en són tres: una pinça elèctrica al quadre general de l'habitació que mesura la potència en Watts, i dos sensors d'aigua a les canonades principal de l'habitació, que mesuren els cabals en litres tant d'aigua freda com d'aigua calenta.

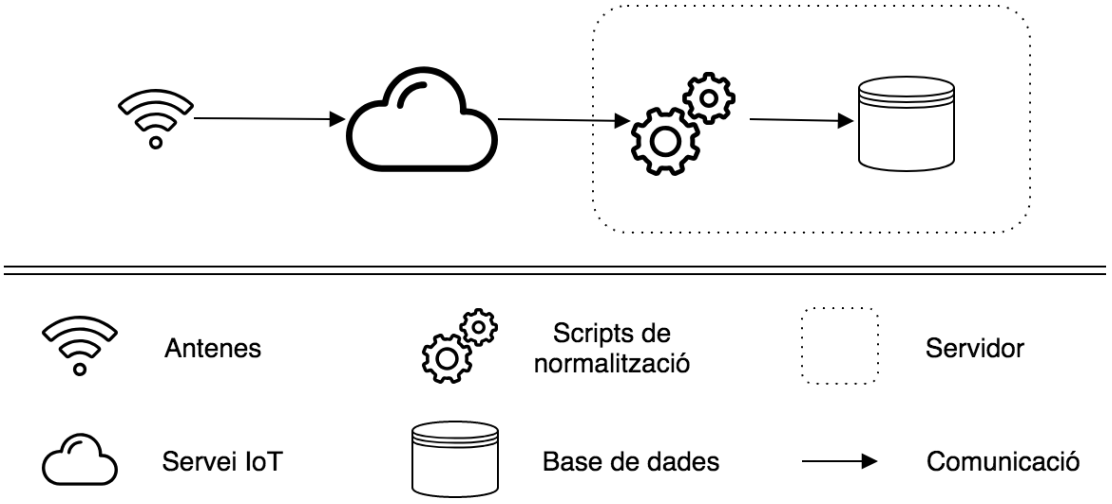


Figura 2. Esquema IoT

Aquests sensors estan directament connectats a nodes que incorporen una antena. Els nodes, en cas que els sensors hagin detectat un consum, enviaran una trama de dades a través de la xarxa de comunicació usada. Actualment, s'utilitza la xarxa oferta per Sigfox.

Les trames són captades pel cloud de Sigfox, i enviades, a través del callback establert, al servidor de GreenCustomers usant els scripts de normalització.

Seguidament, un cop el servidor rep la trama, aquest ha de normalitzar els valors. Durant el procés, es realitzen un seguit de comprovacions per detectar si la trama és correcta, o conté algun error, com per exemple: conèixer si s'ha perdut una trama o els valors estan fora del rang previst que el sensor és capaç de mesurar. Un cop s'ha tractat la informació de forma exitosa, aquesta s'atribueix a la zona on s'ha generat. Al final de la normalització, podem saber que, per exemple: "En l'habitació 101, de l'hotel 'Plaza', a les 12 hores d'avui, hi ha hagut un consum de 50 Watts i 15 litres".

Finalment, a través del portal o del backoffice, es pot accedir a aquesta informació en forma de gràfics i taules. Per exemple es poden consultar els històrics, que informen quina quantitat s'ha gastat en un període de temps: el total i en cada període de 30 minuts:

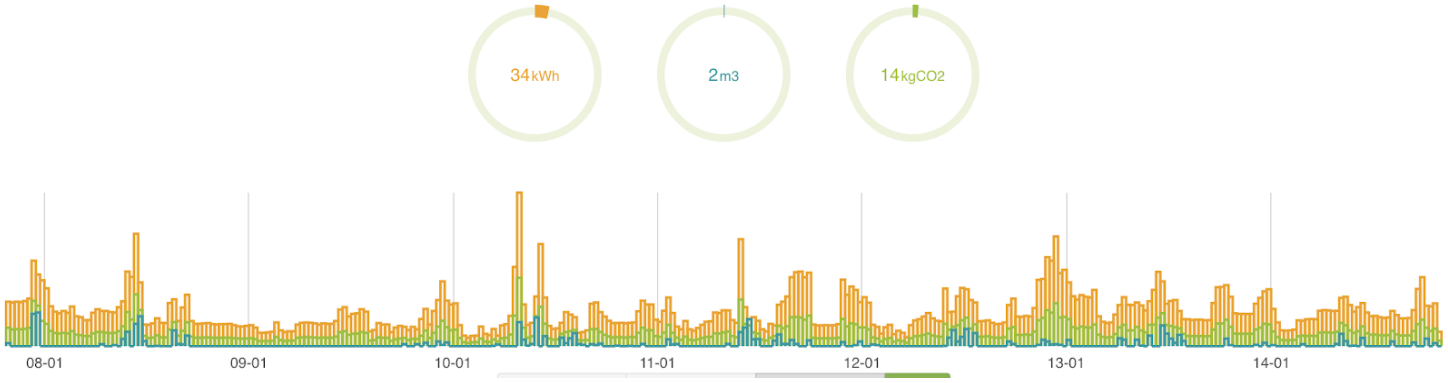


Figura 3. Exemple d'històric de dades

Un altre exemple és una taula de mètriques en què, donat un període de temps, calcula quin ha sigut el pic, la vall i la mitjana en el consum d'energia i aigua, i la comparació d'aquests valors amb l'últim any:

Energy	Max Energy (Wh)	Average Energy (Wh)	Min Energy (Wh)
Selected dates	3057	2256	755
Last year	4302	2140	1034
Difference	-28%	5%	-26%

Total Water	Max Water (l)	Average Water (l)	Min Water (l)
Selected dates	219	138	23
Last year	444	101	23
Difference	-50%	36%	0%

Figura 4. Exemple de taules de mètriques

3 Anàlisi del Software

En aquest capítol es detallaran els problemes i debilitats que presenta l'estat actual del backoffice, les bases de dades i els scripts de normalització. Un problema, segons el següent anàlisi, és un defecte en l'arquitectura o en l'aplicació que impedeix al sistema, totalment o amb un gran nombre de dificultats, créixer i/o evolucionar. Una debilitat, és característiques que si el creixement o evolució però en limiten les possibilitats.

3.1 Problemes

En el procés d'anàlisi es van detectar tres problemes que perjudiquen les aplicacions del sistema. Aquests problemes són els que es voldran resoldre de forma prioritària en el nou disseny del sistema:

Problema 1: Procés de normalització de les dades

El primer i més gran dels problemes és l'actual sistema de normalització. Com s'ha explicat tant en l'estat de l'arquitectura actual com del resum de funcionament, el sistema rep dades recollides per sensors que han de ser normalitzades i emmagatzemades correctament. La part clau del procés de normalització es troba en funcions i disparadors en les pròpies bases de dades. Un canvi en aquest sistema de normalització implica modificar la capa de dades del sistema. La dependència provoca greus limitacions en futures millores de les funcionalitats actuals i en possibles noves característiques. Un dels exemples és l'agregació de nou tipus de sensors, com els de temperatura o pressió, per exemple. En aquest cas, s'haurien de revisar els disparadors en les bases de dades.

Problema 2: Multiplicitat de bases de dades

El segon problema identificat, que es relaciona amb el primer, és que, degut a l'existència d'una base de dades per hotel, part de les funcions SQL es troben replicades a cada base de dades. Un canvi en l'script implica de forma directa haver de copiar el nou codi tants cops com hotels es trobin en el sistema. No només augmenta el temps necessari per aplicar els nous canvis, sinó que també pot induir a errors més fàcilment.

Problema 3: Tecnologia obsoleta

El tercer problema és la tecnologia utilitzada pel backoffice i el portal: a l'inici es va triar *php* amb el framework MVC de *CodeIgniter 2*, i per al *back-end* i per al *front-end* no es va fer servir cap *framework*. CodeIgniter és un *framework php* per a *back-end*s que es basa en l'estructura MVC (Model-Vista-Controlador). La versió usada és la 2.2.6. Aquesta va ser publicada l'Octubre del 2015 [4]. Tot i les seves capacitats, actualment està molt desactualitzada tecnològicament respecte altres *frameworks php* o d'altres lleguatges com *Django* (fet amb *python*) o *Ruby On Rails* (fet amb *ruby*). En el cas del *front-end*, el fet de no utilitzar un framework de javascript empitjora tant l'eficiència com l'experiència d'usuari. Les pàgines s'han de recarregar en cada acció realitzada. Per a contrarestar els efectes, es va afegir una capa de javascript per a intentar dinamitzar les accions. La capa però és complexa a causa de la lògica necessària per a simular el comportament d'un *framework*. En conseqüència, la capa de javascript no facilita la introducció de canvis o nous elements de control.

3.2 Debilitats

A més dels problemes detectats, existeixen debilitats que necessiten ser corregides. Són debilitats ja que per si mateixes no causen problemes en l'escalabilitat o en la modificació del comportament del backoffice o del portal.

Debilitat 1: Connexió IoT-Servidor

La primera és la connexió del cloud del servei IoT directa amb scripts de normalització independents del backoffice. Aquesta independència es pot observar en l'esquema de l'arquitectura actual en el punt 2.2 del document. Amb aquesta independència perdem l'accés a la lògica que pot oferir el backoffice: Ús de codi existent i un control d'errors més efectiu.

Debilitat 2: Back-end Portal

La segona debilitat és l'existència d'un back-end propi per al portal. El portal és una aplicació en què la seva principal funció és el mostreig de gràfics i taules. De la mateixa manera que els scripts de normalització, gran part del codi que actualment té el back-end del portal, pot ser substituït per codi del backoffice.

Debilitat 3: Informacó no centralitzades

La tercera debilitat sorgeix perquè el sistema no emmagatzema el 100% de la informació. Actualment s'utilitzen fulls de càlcul per tenir un registre dels tipus de nodes que es disposen. Un nou sistema hauria de contemplar ser el responsable de guardar-ho per a garantir la centralitat de les dades.

Debilitat 4: Àrees complexes

El sistema actual permet crear representacions d'espais físics, però és impossible poder crear subzones per a obtenir detalls més precisos com, per exemple: ser capaç de detectar els consums de la climatització d'una habitació a part del conjunt global d'aquesta.

Debilitat 5: Nodes limitats

Actualment, i des de l'inici del projecte, s'utilitzen nodes amb tres interfícies d'entrada i/o sortida. Per aquest motiu, en el sistema només contempla la possibilitat que un node tingui fins a tres sensors connectats. El nou sistema, ha de permetre definir nodes que puguin tenir un nombre indeterminat de sensors connectats.

4 Definició dels requisits

Els problemes i debilitats detectades en l'anàlisi del capítol anterior afecten tant l'arquitectura del sistema com a funcionalitats en els serveis. A causa d'aquestes afectacions, s'han de tenir en compte els nous requisits, tant funcionals com no funcionals, que les solucions que es plantegin hagin de complir per a resoldre els problemes amb la màxima cobertura possible.

4.1 Requisits no funcionals

Per a assolir una arquitectura amb una vida útil mirant a mitjà/llarg termini, s'han plantejat els següents requisits no funcionals.

Els requisits no funcionals han estat categoritzats segons l'estàndard descrit en l'ISO 9126-1 i posteriorment actualitzats en l'ISO/IEC 25010:2011. [5]

Requisits sobre la funcionalitat

Requisit de la completesa de les funcionalitats	
Descripció	Grau en què un conjunt de funcionalitats cobreix les tasques i objectius especificats.
Justificació	Tots els objectius marcats i tasques descrites han de poder dur-se a terme amb alguna de les funcionalitats del sistema.
Satisfacció	Tots els casos d'ús han d'estar reflectits en el sistema.

Requisit de l'apropiativitat de les funcionalitats	
Descripció	Grau en què una funcionalitat facilita la realització satisfactòria d'una tasca concreta.
Justificació	Les funcionalitats han de ser plantejades de manera que l'usuari pugui seguir una successió de passos clars, excloent-hi passos innecessaris.
Satisfacció	Qualsevol tasca ha de poder realitzar-se amb un nombre baix de passos.

Requisits d'eficiència

Requisit de temps de resposta	
Descripció	Grau en què els temps de resposta i processament del sistema, s'adeqüen als requeriments.
Justificació	Una acció ha de respondre amb un temps apropiat a la complexitat de la tasca que realitza.
Satisfacció	Les tasques bàsiques no han de durar més de 5 segons tenint en compte una connexió a internet bàsica.

Requisits d’usabilitat

Requisit d'aprenentatge autònom	
Descripció	Grau en què el sistema pot ser utilitzar pels usuaris alhora que aquests aprenen a fer-lo servir amb efectivitat i eficiència.
Justificació	El sistema ha de ser intuïtiu perquè un usuari pugui utilitzar-lo des d’un inici sense formació i sigui fàcil recordar com realitzar les mateixes accions en el futur.
Satisfacció	Tots els menús han d’estar indicats amb el contingut corresponent i els formularis han d’estructurar-se amb un ordre lògic.

Requisit de protecció a l’usuari	
Descripció	Grau en què el sistema protegeix a un usuari de cometre errors durant la realització d’una tasca.
Justificació	El sistema ha de guiar a l’usuari per tal de reduir i/o eliminar errors innecessaris i/o evitables.
Satisfacció	En qualsevol cas d’error s’ha d’explicar a l’usuari quin ha estat l’error produït i com evitar-lo novament en el futur.

Requisit d’estètica de la interfície	
Descripció	Grau en què una interfície és agradable als usuaris
Justificació	La interfície ha de ser neta i ha d’estar ben estructurada
Satisfacció	La interfície ha de mostrar la informació necessària en cada moment i ha d’evitar distreure a l’usuari

Requisits de fiabilitat

Requisit de tolerància d’error	
Descripció	Grau en què un programa opera en normalitat en cas d’error en el hardware o software
Justificació	El sistema ha de programat per a resistir qualsevol error que pugui produir-se.
Satisfacció	Tots els casos d’ús han de contemplar tots els possibles escenaris que no garanteixin la satisfactòria realització de la tasca que descriuen.

Requisits de seguretat

Requisit de confidencialitat	
Descripció	Grau en què un programa permet l’accés a les dades només a aquelles persones que estan autoritzades.
Justificació	Cada usuari ha de poder guardar dades privades.
Satisfacció	El sistema farà comprovacions de permisos abans d’autoritzar l’acció que es vol realitzar.

Requisit de responsabilitat	
Descripció	Grau en què les accions d'una entitat poden ser rastrejades únicament a la mateixa entitat
Justificació	Cada entitat ha de contenir un atribut únic a la classe per tal de poder-la identificar en qualsevol moment.
Satisfacció	Totes les classes representades a la base de dades contindran una clau primària única

Requisits de manteniment

Requisit de modularitat	
Descripció	Grau en què un sistema està compost de mòduls que, en cas de modificació d'un d'ells, la resta no es veuen perjudicats o afectats.
Justificació	Cada component del sistema ha d'actuar sobre l'entitat per la qual s'ha dissenyat.
Satisfacció	Cada component només s'encarregarà de tasques específiques amb clars paràmetres d'entrada i sortida.

Requisit de canviabilitat	
Descripció	Grau en què un sistema pot ser modificat sense introduir defectes.
Justificació	El sistema ha de poder evolucionar.
Satisfacció	La introducció de nous canvis no deriva en una cascada de modificacions en cadena per adaptar el sistema a un nou estat.

5 Anàlisi de solucions

En aquest capítol es presentaran diferents alternatives per a resoldre els problemes i debilitats detectades en el capítol anterior. S'hauran de tenir en compte l'impacte que poden tenir les solucions en el sistema actual, però també el cost que aquestes comporten i si permeten la re-utilització de recursos existents.

5.1 Propostes de solucions

5.1.1 Problema 1: Procés de normalització

Alternativa única: Extreure els triggers

L'única solució proposada per resoldre el problema és reconvertir el codi dels *triggers* i les funcions de les bases de dades en scripts externs. D'aquesta manera alliberem la capa de dades de lògica computacional i permetem poder realitzar canvis en els serveis o afegir components nous sense haver de propagar els canvis a l'estructura de la base de dades.

5.1.2 Problema 2: Multiplicitat de bases de dades

Alternativa 1: Base de dades unificada

El primer que es proposa és la unificació de totes les dades en una sola base de dades. Proposant aquesta solució s'elimina la multiplicitat de bases de dades existents que trobem on hi ha una per hotel. Tot i això, amb aquest resultat es crearia una taula on hi anirien totes les dades dels consums detectats pels sensors. Tenint en compte que cada habitació disposa de dos sensors, com a mínim, un d'aigua i un d'elèctric, els quals envien una dada cada 30 min: obtenim que una habitació genera més de 17.500 registres en un any. Només que el sistema contingui 30 habitacions actives, aquestes acumularien 525.000 de registres l'any. En un projecte que s'espera que escali, el cost d'accés creix ràpidament.

Fortaleses:

- Elimina la multiplicitat de les bases de dades.
- Redueix la complexitat de l'estructura.

Debilitats:

- Creació d'una taula amb un alt creixement del nombre files.
- S'ha de fer una migració important d'informació.

Alternativa 2: SQL + NoSQL

La idea és reestructurar el contingut de forma que tota la informació a excepció de les dades dels consums s'emmagatzemin en una única base de dades SQL. Per una altra banda, es

planteja migrar les dades de consums a un tipus de base de dades NoSQL. S'ha de valorar si hi ha algun tipus de base de dades no relacional que s'adapti al format de les dades que guardem.

Fortaleses:
- Elimina la multiplicitat de les bases de dades.
- Redueix la complexitat de l'estructura.
- Permet escalar més fàcilment el volum de dades de consums.

Debilitats:
- S'ha de fer una migració important d'informació.
- S'ha d'afegir un adaptador capaç de comunicar-se amb la nova base de dades NoSQL.

Alternativa 3: Reestructuració parcial

L'objectiu és poder centralitzar en una sola base de dades tota la informació a excepció de les dades de consums, les quals romandran en bases de dades separades. D'aquesta manera es redueix la complexitat de l'estructura, alhora que s'evita haver de moure els registres acumulats fins al moment.

Fortaleses:
- Redueix la complexitat de l'estructura.
- Evita fer una migració important d'informació

Debilitats:
- Es manté, de forma parcial, la multiplicitat de les bases de dades.

5.1.3 Problema 3: Tecnologia obsoleta

Alternativa 1: Utilitzar una nova versió del framework actual

El framework que s'utilitza, *CodeIgniter*, té una nova versió. Utilitzant-la assegurem el re-aprofitament de quasi tot el codi. Tot i això, la nova versió no aporta moltes millores respecte a la versió actual, i ja s'està treballant en el desenvolupament de la versió 4. Fet que pot deixar l'aplicació amb una tecnologia enrederida en un període relativament curt de temps. Addicionalment, el *framework* no permet una integració senzilla amb *frameworks javascript* per al desenvolupament del *frontend* de l'aplicació.

Fortaleses:
- Permet la reutilització de quasi tot el codi
- És conegut per l'equip

Debilitats:

- No presenta millores substancials.
- Difícil integració amb *frameworks javascript*.

Alternativa 2: Ruby on Rails + ReactJs

La proposta és canviar la tecnologia per l'ús de *Ruby on Rails* 5 (RoR) [6]. Aquest *framework* escrit en *ruby*. És un *framework* amb una comunitat activa que s'adapta a les noves tecnologies i, a més, ja és conegut tant per l'equip com pel desenvolupador. Aquest incorpora eines integrades en el *core* per a l'ajuda al desenvolupament. Per exemple *ActiveStorage*: una API interna per a l'emmagatzematge de documents; o *ActiveCable*: una API per a l'ús de WebSockets/Subscriptors entre el servidor i el client. A més, RoR utilitza el sistema de gemmes de *ruby*. Una 'gemma' és una llibreria escrita en *ruby*. Entre les més usades està *Devise*. *Devise* és una llibreria que millora i amplia el sistema de gestió d'usuaris i sessions, i garanteix la seguretat amb encriptació i l'ús de *tokens* [7]. A més a més, RoR, amb la gema react-rails [8], s'integra amb el framework de javascript ReactJs [9].

Fortaleses:

- Ús d'un framework més nou amb més facilitats.
- S'integra amb *frameworks* javascript.
- És conegut per l'equip.

Debilitats:

- S'ha de traduir el codi.

5.1.4 Debilitat 1: Connexió IoT-Servidor

Alternativa 1: Mantenir els scripts separats

La primera proposta és mantenir els scripts de normalització separats de la resta de serveis (*backoffice* i portal). D'aquesta manera podem crear un script que utilitzi només els recursos que siguin realment necessaris, fent èmfasi en l'eficiència d'aquests. Tot i això, hem de ser més curosos en la seguretat i en la persistència de les dades en el procés, ja que sense un *framework* és el desenvolupador el que ho ha d'assegurar.

Fortaleses:

- Es mantenen els serveis separats.
- Es pot aprofundir en l'eficiència.

Debilitats:

- No s'aprofiten els recursos del backoffice.

Alternativa 2: Integrar els scripts en el backoffice

La segona proposta és integrar el procés de normalització de les dades en el backoffice. Per fer-ho s'habilitarà un *end-point* com a *WebHook* del servei IoT *[Per a més informació sobre els WebHooks, consultar l'apèndix 1]*. Amb conseqüència, tots els recursos del backoffice estan disponibles: accés fàcil i directe als models i bases de dades, accés a les interfícies internes com SMTP o WebSockets, i protocols de seguretat del propi *framework*.

Fortaleses:

- S'aprofiten els recursos del backoffice.

Debilitats:

- No es manté la separació de serveis.
- S'utilitzen més recursos que actualment.

5.1.5 Debilitat 2: Back-end Portal

Alternativa 1: Mantenir el portal com està

La primera proposta és mantenir el portal com una aplicació web completa amb *frontend* i *backend*: amb codi client i codi servidor. Es pot plantejar la renovació del back-end del portal com a millora de futur del projecte. Però no es resoldrà el tema de fons que és que, tot i mantenir els serveis completament separats, no es podrà reutilitzar els recursos que el back-office disposa.

Fortaleses:

- Es mantenen els serveis clarament separats.

Debilitats:

- No s'aprofiten els recursos del backoffice.

Alternativa 2: Absorbir el backend del portal

La proposta és eliminar completament el backend del portal web. Convertir el portal en una *"Front-End Single-Page Application"*. Això significa crear una aplicació amb codi que s'executi només al client: html, css i javascript; i que consti d'una sola pàgina. Per a dur a terme la solució, es prepararà una API en el back-office com a punt de connexió entre el portal i les dades.

Fortaleses:

- S'aprofiten els recursos del backoffice
- Se simplifica la complexitat de la infraestructura-

Debilitats:

- Es crea una jerarquia de dependència de serveis.

5.1.6 Debilitat 3: Informació no centralitzada

Alternativa única: Incorporar la informació

L'única proposta que permet l'eliminació d'aquesta debilitat és senzillament modelitzar els tipus de node en el sistema i afegir la interfície i lògica necessàries per a la seva gestió i administració.

5.1.7 Debilitat 4: Àrees complexes

Alternativa única: Jerarquització recursiva

Es proposa que les àrees es puguin jerarquitzar de forma recursiva. D'aquesta manera, la jerarquia es pot estendre els nivells que facin falta i s'obté un marge de maniobra major per adaptar-se a cada situació amb més precisió.

5.1.8 Debilitat 5: Nodes limitats

Alternativa única: Actualitzar les relacions

Es proposa canviar les relacions entre les classes sensor, node i àrea per tal que recullin la possibilitat d'assignar un nombre indeterminat de sensors a un node depenent del nombre d'interfícies d'aquest.

5.2 Solucions triades

Un cop conegudes les alternatives de solucions proposades per a resoldre cada problema i debilitat analitzada s'han d'escollir aquelles que garanteixen la màxima cobertura del problema/debilitat i tinguin una bona proporció entre el benefici que aporten i el cost de realització. En les següents taules es mostren les alternatives per a cada problema i debilitat, i s'ha ressaltat en negreta la solució triada en cada cas.

Problema		Solucions	
1	Procés de normalització	1	**Extreure els *triggers***
2	Multiplicitat de bases de dades	1	Base de dades unificada
		2	SLQ + NoSQL
		3	**Reestructuració parcial**
3	Tecnologia obsoleta	1	Nova versió del *framework* actual
		2	**Ruby on Rails + ReactJs**

Taula 1. Resum problemes i alternatives

Debilitat		Solucions	
1	Connexió IoT-Servidor	1	Mantenir els *scripts* separats
		2	**Integrar els *scripts* en el *backoffice***
2	Back-end Portal	1	Mantenir el portal
		2	**Absorbir el back-end del portal**
3	Informació no centralitzada	1	**Incorporar la informació**
4	Àrees complexes	1	**Jerarquització recursiva**
5	Nodes limitats	1	**Actualitzar relacions**

Taula 2. Resum debilitats i alternatives

5.2.1 Problema 1: Procés de normalització

Pel primer problema, ja que només hi ha una alternativa a resoldre'l, es realitzarà el que aquesta proposa: Treure tota la lògica computacional de les bases de dades. Es garanteix la independència de capes (dades-controladors) i s'augmenta la independència dels serveis.

5.2.2 Problema 2: Multiplicitat de bases de dades

Per fer front al segon problema s'escull l'alternativa 3. La tercera alternativa manté la multiplicitat de les bases de dades, però redueix clarament la complexitat de l'estructura. Només es guardaran els registres de les dades de consum, la resta d'informació se centralitza en una sola base de dades. A més, evita haver de crear una única taula amb un potencial de creixement elevat, i, en comparació amb la segona proposta d'usar NoSQL, s'evita també haver de fer migracions sobre aquests registres.

5.2.3 Problema 3: Tecnologia obsoleta

Pel tercer problema, es realitzarà la proposta 2. Tot i que utilitzar un *framework* diferent comporta més feina, ja que significa traduir el codi, els beneficis que RoR aporta té un pes més elevat que mantenir CodeIgniter encara que sigui amb l'última versió.

5.2.4 Debilitat 1: Connexió IoT-Servidor

La solució que s'aplica per a resoldre la debilitat és la descrita en la segona alternativa. Es prefereix integrar els scripts en el *backoffice*. L'accés als recursos, eines i lògica que el *backoffice* disposa permetrà simplificar la complexitat dels scripts de normalització.

5.2.5 Debilitat 2: Back-End Portal

Per a millorar la segona debilitat, s'escollirà la segona alternativa. Fent que el portal rebi les dades a través del backoffice, usant una api, fem que es redueixi el nombre d'elements en la infraestructura amb accés directe a les dades, i es té un major control sobre aquests accessos.

5.2.6 Debilitat 3: Informació no centralitzada

Per aquesta debilitat només existeix una solució per a resoldre-la. El nou sistema haurà d'incloure els tipus de nodes disponibles i la seva informació tècnica bàsica que permeti comprendre com és el node.

5.2.7 Debilitat 4: Àrees complexes

Per a permetre una jerarquia flexible, s'adaptarà el sistema fent que una àrea pugui contenir subàrees recursivament. S'incrementa la lògica de control per tal d'evitar la creació de cicles, però no es dificulta l'estructura de dades.

5.2.8 Debilitat 5: Nodes limitats

Es canviarà l'estructura de les dades de nodes i sensors per a permetre l'assignació de més de tres sensors a un node específic. Això incrementarà tant la complexitat de les dades com la lògica de control.

6. Metodologia

6.1 Metodologia de Treball

GreenCustomers, al ser una StartUp, treballa amb MVP en els seus projectes segons estableix el mètode Lean StartUp [10]. Això vol dir que aquest se centrarà només en el que es cregui que aporti valor real. En Lean StartUp s'utilitza la metodologia de "posada a producció continua". És a dir, tot el codi nou s'afegeix a l'entorn de producció un cop ha estat completat. El que redueix el temps de cicle d'entrega del producte [10].

En aquest treball, però, s'ha optat per seguir una metodologia Àgil. Aquesta metodologia es basa en el desenvolupament de software amb equips interdisciplinaris i amb la col·laboració directa dels clients. S'avoca per seguir una planificació que s'adapti i evolucioni als canvis, i permetre el desplegament continu de la solució [11]. Aquesta metodologia va ser inicialment estandarditzada per l'*Agile Manifesto* [12], el qual consta de 12 principis i de quatre valors bàsics. Aquests últims són els descrits a continuació:

- **Iteracions i *Sprints***: Es tracta de dividir en parts més petites el desenvolupament d'un software en què cada una passarà per totes les fases: planificació, disseny, desenvolupament, pla de proves, revisió i desplegament.

- **Software que funcioni**. L'objectiu és crear nou software per sobre de documentació.

- **Col·laboració amb el client**. Un contracte és important però ho és més saber les seves necessitats.

- **Resposta al canvi**. És més important respondre als canvis de forma encertada i no aferrar-se a la planificació inicial.

De forma resumida: quan es desenvolupa un projecte o producte en metodologia Àgil, aquest se segmenta en blocs que per si sols aporten un valor. El treball realitzat en cada bloc s'anomena sprint, i en cada un, es realitzen els passos que es realitzen en una metodologia de cascada: planificació, disseny, desenvolupament, pla de proves, entrega. Actuant amb aquest mètode, no existeix una planificació global ni una data final d'entrega, pel que sempre s'hi poden introduir canvis, ja sigui per petició del client donats nous requisits, o a causa de factors no previstos que obstaculitzen el desenvolupament.

El motiu que s'escull la metodologia Àgil, per sobre de *Lean* és que el projecte necessita un mètode de validació previ a la posada a prova. En *Lean*, es decideix si una eina o funcionalitat realment aporten un valor si aquesta és utilitzada pels usuaris. En Àgil, una eina s'afegeix al sistema si aquesta ha estat demanada pel client i ha estat prèviament avaluada.

6.2 Eines de seguiment

Per dur a terme el seguiment del projecte, i el control de versions, es farà servir un repositori Git en un servidor de Bitbucket. Per a interactuar amb aquest, s'utilitzarà l'eina SourceTree. L'eina ofereix una interfície gràfica per a la gestió de repositoris Git siguin de Bitbucket o GitHub. D'aquesta manera, es pot observar més clara i fàcilment l'evolució del projecte.

A més, a final de cada sprint, es realitzarà una reunió per avaluar com ha evolucionat aquest i preparar el següent per veure si s'han de fer canvis sobre la planificació inicial en cas d'obstacles.

6.3 Metodologia de Validació

Per a la contínua validació del projecte, es realitzaran reunions amb els responsables de GreenCustomers que serviran per revisar la feina realitzada no només l'*sprint* que acaba i el que ha de començar, sinó també l'estat general del treball.

Per al que fa a la memòria, les entregues realitzades durant el curs de Gestió de Projectes, GEP, seran la base de la documentació del projecte. Un cop GEP hagi conclòs, les reunions amb la tutora del projecte, s'utilitzaran per a comprovar l'estat d'aquesta i els canvis que s'haurien d'introduir per a la millora en la següent entrega.

7 Planificació i Gestió

En aquest capítol es definirà la planificació inicial del projecte i quina serà la gestió econòmica i dels recursos que s'utilitzaran. Inicialment es detallaran condicions que s'han de tenir en compte, seguidament s'explicarà a escala d'sprint les tasques que es duran a terme i el cost que representa dur-les a terme, i al final hi haurà una visió global de la planificació i del pressupost total.

7.1 Consideracions Inicials

7.1.1 Calendari

Aquest projecte es vol desenvolupar durant el mes d'agost del 2018 i el transcurs del quadrimestre de tardor del curs 2018/2019. Segon la normativa de la facultat, el projecte té un pes de 15 crèdits amb una càrrega de treball de 25 hores per crèdit. El total d'aquest, per tant, ha de sumar 375 hores de treball.

Cal tenir en compte les alternatives a possibles desviacions que poden originar-se per culpa d'imprevistos en la planificació. Per fer front a l'amenaça, s'afegiran 25 hores addicionals a les del còmput de crèdits previst per la universitat. Augmentant fins a 400 hores el total del projecte. A més, es comptabilitzaran a l'alça el pes d'algunes tasques amb més complexitat per a garantir-ne el temps i dedicació que requereixen. En cas que els endarreriments superin el marge establert, s'hauran d'afegir hores extres al calendari per a contrarestar els efectes causats. I en casos extrems, on la planificació es vegi greument afectada, s'haurà de canviar la planificació: reordenar tasques, o com a última decisió eliminar-ne alguna.

Per a complir el calendari s'han repartit les hores en 7 sprints els quals es repartiran de la següent forma i pes:

Sprint	Dates	Hores dedicades
1	20 d'Agost - 9 de Setembre	57
2	10 de Setembre - 30 de Setembre	57
3	1 d'Octubre - 21 d'Octubre	57
4	22 d'Octobre - 11 de Novembre	57
5	12 de Novembre - 2 de Desembre	57
6	3 de Desembre - 23 de Desembre	57
7	24 de Desembre - 13 de Gener	58
	Total:	400

Taula 3. Planificació dels sprints

En aquestes previsions ja es tenen en compte la impossibilitat de dedicar hores els dimarts i dijous durant el transcurs de les classes del quadrimestre de tardor del curs actual a causa de l'horari de feina i classes, i també, les celebracions de Nadal durant els primers 4 dies del setè sprint.

Aquesta distribució està pensada perquè es dediquin 19h setmanals repartides en 5 dies:

Dilluns	Dimarts	Dimecres	Dijous	Divendres	Dissabte	Diumenge	Total
4h	0h	4h	0h	3h	4h	4h	19h

Taula 4. Horari setmanal*

**Aquest horari és el previst durant el transcurs del període de classes del Q1 del curs 2018-19 (10 de Setembre 2018 - 21 de Desembre 2018)*

7.1.2 Recursos

Per dur a terme el treball, es requeriran recursos tan físics, com programari, i serveis contractats a tercers. Concretament seran necessaris les següents eines i material:

- Ordinador portàtil: per a poder treballar des de qualsevol lloc en cas necessitat.
- Connexió Internet: per a poder tenir accés als serveis cloud i accés a la documentació
- Servidor: necessari per a la demo com a entorn de test.
- *Visual Code Studio* com a editor de codi multillenguatge. [13]
- *Git* com a gestor de versions del codi del projecte. [14]
- *Docker* com a gestor de contenidors per encapsular el projecte i millorar la compatibilitat del software entre els diferents entorns de *test* i producció. [15]
- Navegador web per provar l'aplicació web durant el desenvolupament i la fase de prova.
- *Postman* com a entorn de proves de la *API REST*. [16]
- *Google Drive* com a *Cloud Service* per a emmagatzemar i desenvolupar la documentació del projecte. [17]

- *Tom's planner* com a eina utilitzada per a la creació dels diagrames de GANTT presents en el document. [18]

7.1.3 Identificació dels costos

Els costos d'un projecte, es divideixen en dos: els costos directes, que són aquells que afecten directament a la realització del projecte; i els costos indirectes, que s'atribueixen a aconseguir i/o mantenir les condicions i requisits necessaris per a poder desenvolupar el projecte. [19]

Dins dels costos directes, hem de destacar-ne dos subgrups: els costos de capital humà i els costos de material.

Els costos humans d'un projecte, són els que s'atribueixen al valor de la dedicació de totes les persones involucrades. En altres paraules: el sou atribuït al rol que realitza cada actor implicat. En aquest projecte, recuperant els actors implicats definits prèviament en aquest document, s'ha de tenir en compte el cost del CEO i CIO de GreenCustomers i també del desenvolupador. Segons la política pròpia de GreenCustomers, la base salarial s'estableix de la següent forma:

Rol	Salari (€/h)
CEO	10
CIO	10
Desenvolupador	8

Taula 5. Salaris del rols implicats

Els costos de material són els del preu dels recursos i serveis esmentats en el punt anterior. Ja que algunes eines tenen una vida útil més llarga que la durada del projecte, només es computen els costos proporcional de l'ús únicament necessari en aquest projecte.

Per altra banda, els costos indirectes de qualsevol projecte són principalment les despeses del manteniment de l'entorn de treball i oficina. Això inclou el lloguer de l'oficina, les factures d'aigua i llum, així com els serveis de neteja, telefonia i internet. Però també, altres costos que es poden considerar com a indirectes, són els derivats dels desplaçaments dels participants en el projecte.

7.2 Descripció d'*sprints*

En el següent apartat es descriuran els sprints que es realitzaran per a completar el projecte. Cada sprint constarà amb una descripció de les tasques que es duran a terme amb el diagrama de GANTT corresponent i el càlcul dels costos humans que implicarà la realització de l'sprint.

En la planificació realitzada, totes les tasques pertanyents a un sprint donat són dependents del *sprint* anterior. A excepció del primer, el qual no té dependències. Totes les tasques s'han organitzat per intentar evitar les dependències dins d'un mateix sprint. En cas de dependència, aquesta serà explicada en *l'sprint* corresponent.

Per ampliar en el contingut, cal aclarir què és un diagrama de GANTT i la llegenda d'aquests, i també com es calcula el cost d'un sprint:

- Un diagrama de GANTT és una eina de planificació per a determinar quines són les actuacions que s'han de dur a terme i quan i amb quin ordre es realitzaran. Els diagrames descrits en cada sprint es regeixen pel següent codi de colors. Cada color representa un tipus de tasca segons els blocs dels objectius marcats en el projecte: gestor de greenroom, sistema IoT i API per als serveis de l'empresa.

Color	Significat
	Especificació i documentació
	Disseny de les interfícies
	WebApp
	API
	Capçalera de l'sprint

Taula 6. Llegenda diagrames de GANTT

- Els costos directes del projecte s'ha de comptabilitzar el cost dels recursos humans com ja s'ha esmentat anteriorment. Tenint en compte el salari de cada un dels actors i dels dies que es dedicaran a cada tasca i el nombre d'hores treballades cada dia. Per tant, a cada tasca i sprint s'ha utilitzat la següent formula:

$$Import\ (€) = \#hores(CEO/CIO) * salari(CEO/CIO) + \#hores(dev.) * salari(dev.)$$

on 'dev.' significa desenvolupador del projecte

7.2.1 Sprint 1 (20 d'Agost - 9 de Setembre): Especificació

Es pretén recopilar els requisits inicials del sistema. Durant aquestes tres setmanes s'estarà amb contacte permanent amb l'empresa per a la recopilació dels requisits que ha de complir el nou projecte. Així com un primer recull de les històries d'usuari i un diagrama de classes. I evidenciar les millores respecte a l'actual.

Les tasques que es duran a terme són quatre i es distribuiran de la següent forma al llarg de l'sprint:

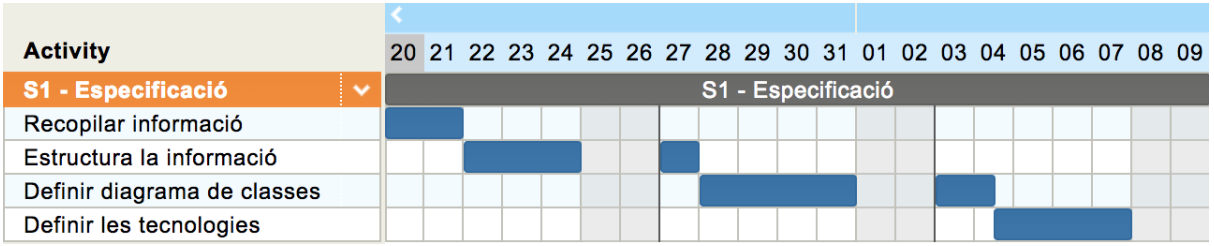


Figura 5. Diagrama de GANTT sprint 1

I el seu cost és el que s'ha descrit en aquest taula:

Activitat	Import			
	Hores Totals Activitat	h. CEO / CIO	h. Dev.	Import (€)
Recopilar Informació	8	8	8	224
Estructura la informació	15	15	15	420
Definir diagrama de classes	21	21	21	588
Definir les tecnologies	13	0	13	104
Total sprint 1	57	44	57	1336

Taula 7. Costos directes sprint 1

7.2.2 Sprint 2 (10 de Setembre - 30 de Setembre): Inici WebApp

Preparació de l'entorn de desenvolupament i inici de la part de l'aplicació web. En aquest sprint es vol començar amb un primer disseny, de les vistes de l'aplicació web, obert a millores i canvis, la gestió d'usuaris i sessions, i amb els models que segueixen un patró d'operacions CRUD: com són els sensors i nodes.

Les tasques que es duran a terme són quatre i es distribuiran de la següent forma al llarg de l'sprint:

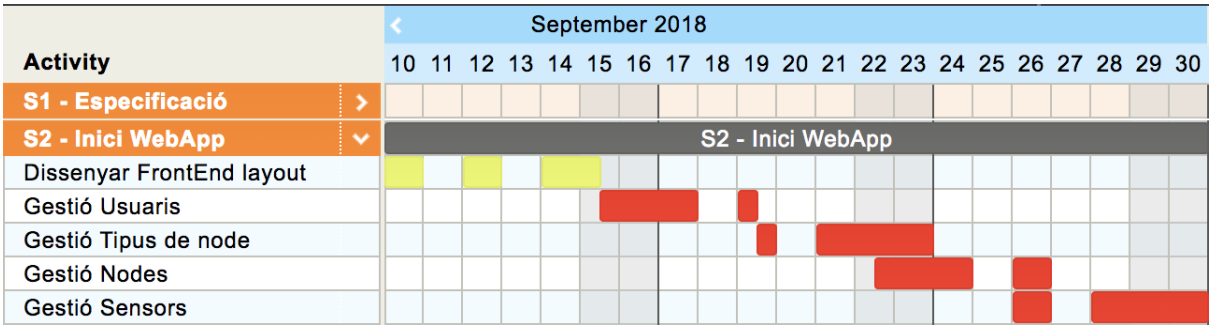


Figura 6. Diagrama de GANTT sprint 2

I el seu cost és el que s'ha descrit en aquest taula:

Activitat	Import			
	Hores Totals Activitat	h. CEO / CIO	h. Dev.	Import (€)
Dissenyar FrontEnd layout	13	0	13	104
Gestió Usuaris	12	0	12	96
Gestió Tipus de node	10	0	10	80
Gestió Nodes	9	0	9	72
Gestió Sensors	13	0	13	104
Total sprint 2	57	0	57	456

Taula 8. Costos directes sprint 2

7.2.3 Sprint 3 (1 d'Octubre - 21 d'Octubre): WebApp:

Seguiment de l'aplicació web: amb els models de Zona i Components. Dos models que segueixen el patró d'operacions CRUD. Se centrarà en la part dels hotels i les àrees i en connectar-ho amb els models creats en l'anterior sprint.

Les tasques que es durant a terme són quatre i es distribuiran de la següent forma al llarg de l'sprint:

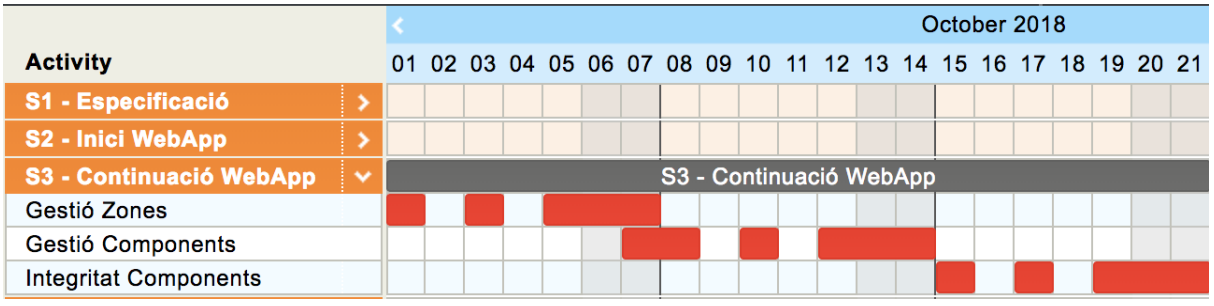


Figura 7. Diagrama de GANTT sprint 3

En l'*sprint*, la tasca "Integritat Components" depèn de la finalització dels panells de gestió definits al *sprint* 2 i als de Zones i Components al mateix sprint. Això es deu, ja que aquest model és la representació de l'associació de diferents models.

I el seu cost és el que s'ha descrit en aquest taula:

Activitat	Import			
	Hores Totals Activitat	h. CEO / CIO	h. Dev.	Import (€)
Gestió Zones	17	0	17	136
Gestió Components	21	0	21	168
Integritat Components	19	0	19	152
Total sprint 3	57	0	57	456

Taula 9. Costos directes sprint 3

7.2.4 Sprint 4 (22 d'Octobre - 11 de Novembre): Registre de les Trames

Creació del registre de trames i estructuració de l'emmagatzematge de les dades rebudes. En aquest sprint es crearà la part dedicada a registrar en un log totes les trames rebudes i preparar l'estructura necessària per guardar les dades normalitzades.

Les tasques que es duran a terme són quatre i es distribuiran de la següent forma al llarg de l'sprint:

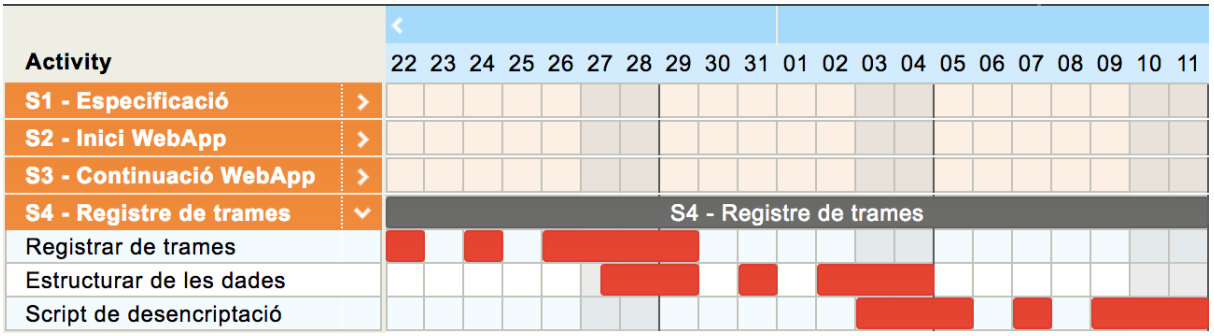


Figura 8. Diagrama de GANTT sprint 4

I el seu cost és el que s'ha descrit en aquest taula:

Activitat	Import			
	Hores Totals Activitat	h. CEO / CIO	h. Dev.	Import (€)
Registrar de trames	18	0	18	144
Estructurar de les dades	16	0	16	128
Script de desencriptació	23	0	23	184
Total sprint 4	57	0	57	456

Taula 10. Costos directes sprint 4

7.2.5 Sprint 5 (12 de Novembre - 2 de Desembre): Control d'alarmes:

Aquesta és una peça important del projecte. Se centrarà en l'script necessari per identificar la font d'origen d'una trama, i la seva correcte normalització. Emmagatzemament de les dades i enregistrament en el log.

Les tasques que es duran a terme són quatre i es distribuiran de la següent forma al llarg de l'sprint:

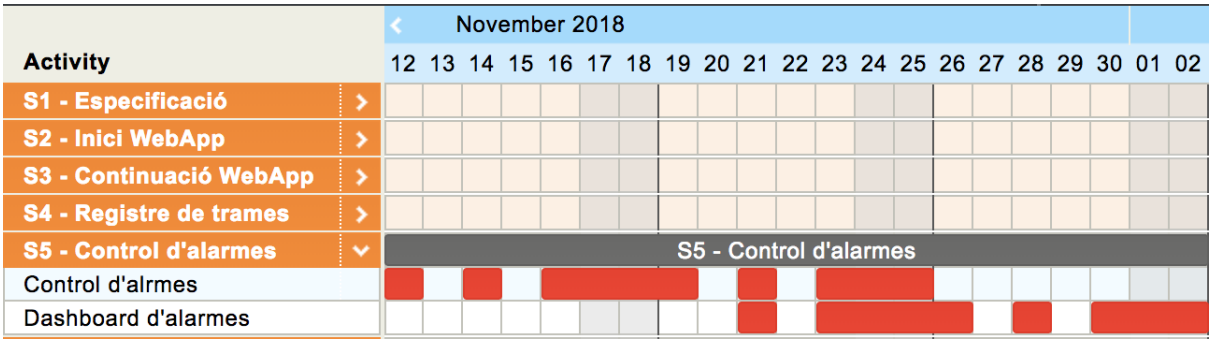


Figura 9. Diagrama de GANTT sprint 5

I el seu cost és el que s'ha descrit en aquest taula:

Activitat	Import			
	Hores Totals Activitat	h. CEO / CIO	h. Dev.	Import (€)
Control d'alarmes	30,5	2	30,5	284
Dashboard d'alarmes	26,5	2	26,5	252
Total sprint 5	57	4	57	536

Taula 11. Costos directes sprint 5

7.2.6 Sprint 6 (3 de Desembre - 23 de Desembre): API

Habilitació de la API per a l'extracció dels consums, estadístiques i mètriques. Es crearà el model de Clients i es farà una API rest perquè aquests puguin tenir accés a l'obtenció de les dades emmagatzemades.

Les tasques que es duran a terme són quatre i es distribuiran de la següent forma al llarg de l'sprint:

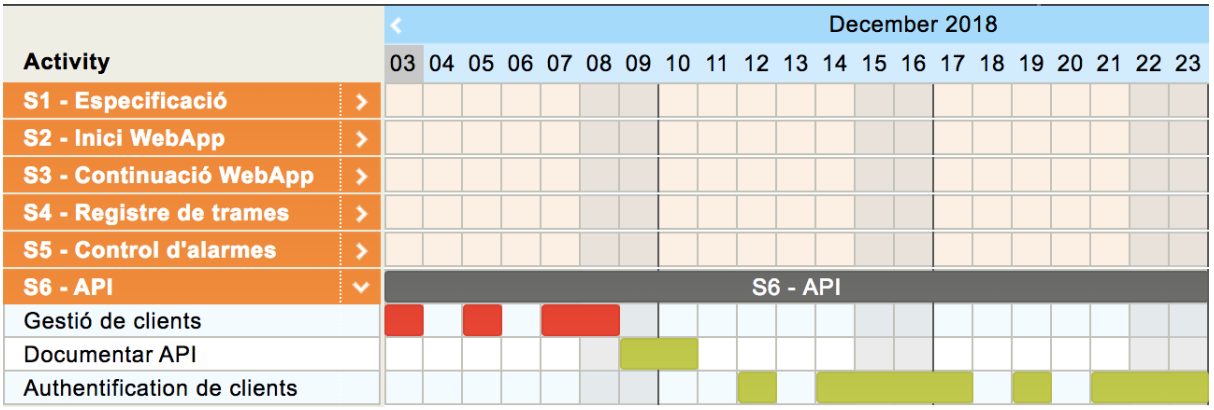


Figura 10. Diagrama de GANTT sprint 6

En l'*sprint*, la tasca "*Authentification* de clients" no es pot començar fins que la *API* estigui documentada, tal com estableix la tasca anterior a aquesta.

I el seu cost és el que s'ha descrit en aquest taula:

Activitat	Import			
	Hores Totals Activitat	h. CEO / CIO	h. Dev.	Import (€)
Gestió de clients	15	0	15	120
Documentar API	8	8	8	224
Authentification de clients	34	0	34	272
Total sprint 6	57	8	57	616

Taula 12. Costos directes sprint 6

7.2.7 Sprint 7 (24 de Desembre - 13 de Gener): Documentació Final

Part final. Aquest sprint es guarda principalment per a l'acabament de la memòria i per tenir un temps de contenció davant de possibles endarrerediments. També per a ja començar a preparar la presentació i arreglar errors tant la memòria com en el codi.

Les tasques que es duran a terme són quatre i es distribuiran de la següent forma al llarg de l'sprint:

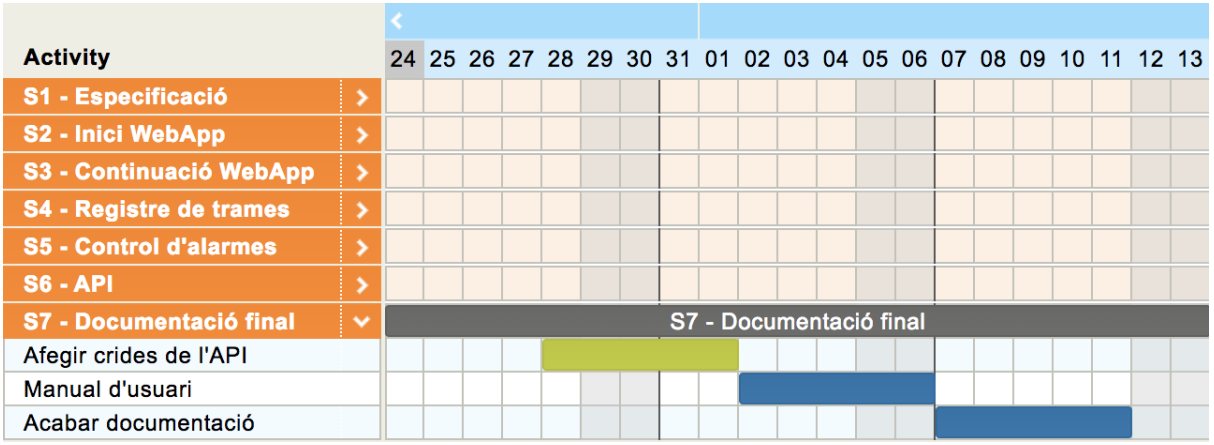


Figura 11. Diagrama de GANTT sprint 7

I el seu cost és el descrit en aquest taula:

Activitat	Import			
	Hores Totals Activitat	h. CEO / CIO	h. Dev.	Import (€)
Afegir crides de l'API	19	0	19	152
Manual d'usuari	19	8	19	312
Acabar documentació	20	0	20	160
Total sprint 7	58	8	58	624

Taula 13. Costos directes sprint 7

7.3 Consideracions finals

7.3.1 Visió global de la planificació

Com ja s'ha dit, el projecte quedarà dividit en 7 sprints de 57 hores cada un, els quals s'hi han atribuït tasques de cada bloc descrites en els objectius del projecte. En la següent taula es mostra aquesta distribució. Addicionalment, en el segon apèndix del document, es pot trobar el diagrama de GANTT complet. [*Apèndix 2. Diagrama de GANTT.*]

Sprint 1	Especificació i requisits inicials
Sprint 2	Funcionalitats bàsiques de l'aplicació web
Sprint 3	
Sprint 4	Connexió de l'aplicació amb el sistema IoT
Sprint 5	
Sprint 6	Desenvolupament de la API
Sprint 7	Finalització de la API i documentació final

Taula 14. Visió global de la planificació

7.3.2 Pressupost final

El pressupost final deriva dels costos de la realització de les tasques planificades, més els costos del material i dels recursos utilitzats durant el desenvolupament del projecte.

Els costos que computen són:

- Els costos directes: del valor de la dedicació de totes les persones involucrades
- Els costos indirectes: provinents dels serveis contractats
- Les amortitzacions: dels recursos materials i del programari usat
- El marge de contingència
- La reserva per imprevistos

Costos directes

En cada *sprint*, s'ha calculat el cost humà esperat per a realitzar cada tasca i el total de *l'sprint*. El total del projecte, per tant, és la suma dels *sprints*. La taula següent recull el total de cada *sprint* i el total del projecte. A més, en l'apèndix del document, es pot trobar la mateixa taula amb el desglossament de costos per tasca amb el còmput d'hores i l'import corresponent. [*Apèndix 3. Estimació costos directes.*]

Sprint	Activitat	Import			
		Hores Totals Activitat	h. CEO / CIO	h. Dev.	Import (€)
1	Total sprint 1	57	44	57	1336
2	Total sprint 2	57	0	57	456
3	Total sprint 3	57	0	57	456
4	Total sprint 4	57	0	57	456
5	Total sprint 5	57	4	57	536
6	Total sprint 6	57	8	57	616
7	Total sprint 7	58	8	58	624
Total		400	64	400	4480

Taula 15. Estimació de costos directes per sprint

Costos indirectes

En aquest bloc, n'hem de tenir dos en compte. El primer i més rellevant és l'oficina. Aquesta consta d'una tarifa mensual tot inclòs que s'aproxima als 650€ mensuals. El segon és el cost del servidor de proves que s'utilitzarà durant el desenvolupament del projecte. Aquest té un cost de 4€ mensuals segons la tarifa contractada vigent amb el proveïdor.

Activitat	Import				
	Import mensual (€)	#mesos	#hores/mes	#horesTFG/mes	Import (€)
Oficina (tot inclòs)	650	6	720	76	411,67
Servidor de proves	4	6	720	76	2,53
Total					415

Taula 16. Estimació costos indirectes

Amortitzacions

Part dels costos indirectes, el que deriven de material físic o programari informàtic, només se'ls computa el preu segon el temps d'ús en el projecte proporcionalment a la seva vida útil. Aquests productes s'amortitzaran durant el període de vida útil. Aquest període, tal com estableix hisenda segons el Reial decret llei 1777/2004, del 30 de juliol: "Reglament de l'Impost de Societats" publicat en el número 189 del B.O.E el 6 d'Agost del 2004: es defineix una vida útil de 3 a 4 anys per a productes hardware i de 2 a 3 anys per a software.[20]

Per a realitzar el càlcul de les amortitzacions, segons la següent fórmula, s'han agafat com a paràmetres aquests valors:

$$Amortització = \frac{cost\ unitari\ (€)}{vida\ útil * dies\ laborables * hores\ treballades\ per\ dia} * hores\ TFG$$

- Vida útil del hardware: 4 anys.
- Vida útil del software: 3 anys.
- Dies laborables a l'any: 220.
- Hores treballades per dia: 4.
- Hores totals del TFG: 400.

Tenint en compte els recursos que s'utilitzaran en aquest projecte, els costos que s'amortitzaran són els descrits en la taula.

Material	Import		
	Cost Unitari (€)	Vida útil (anys)	Amortització (€)
Ordinador	2000	4	227,27
Visual Code Studio	0	3	0
Git	0	3	0
Docker	0	3	0
Postman	0	3	0
Google Drive	0	3	0
Tom's planner	0	3	0
Total			228

Taula 17. Amortitzacions dels recursos

Contingència

Tant en l'abast del projecte, més concretament durant la definició dels possibles obstacles, i en les consideracions inicials de la planificació, en la del calendari, va quedar clar que tot projecte com aquest ha de fer front a qualsevol imprevist.

Per a poder fer-hi front, fixarem un marge de contingència del 10% sobre el valor dels costos estimats, directes i indirectes, i les amortitzacions. Aquest marge de contingència servirà per afrontar l'import que comportaran, per exemple, hores extres treballades les quals sumaran els costos directes dels recursos humans, però també els costos relatius de l'entorn de treball i de les eines utilitzades.

Material	Import
Total Costos Directes	4480,00
Total Costos Indirectes	415
Amortitzacions	228
Total (CD+CI+Am)	5123
Contingència (10%)	513
Total (CD+CI+Am+Cn)	5636

Taula 18. Total Costos, Amortitzacions i Contingència

Imprevistos

Com a seguretat final, davant d'incidències externes no relacionades amb el propi desenvolupament del projecte, però que hi exerceix influència, es reservarà un 10% extra per a pal·liar els efectes negatius ocasionats.

Per tant, el pressupost final quedaria de la següent forma:

Activitat	Import (€)
Total Costos Directes	4480
Total Costos Indirectes	415
Total Amortitzacions	228
Total (CD + CI + Am)	5123
Contingència (10%)	513
Total (CD + CI + Am) + Contingència	5636
Imprevistos (10%)	564
TOTAL	6200

Taula 19. Pressupost FInal

7.3.3 Control de Gestió

Un control de gestió, o de costos, significa comparar el que s'ha estimat en els costos tenint en compte el marge de contingència i el que realment ha tingut lloc. A final de cada sprint, s'avaluarà la càrrega de treball realitzada en comparació a la planificada, i el cost que impliqui aquesta diferència. Per a realitzar aquest còmput se seguiran les regles descrites pels següents indicadors:

- Càrrega de treball: Mesurarà el nombre d'hores que s'han dedicat realment en cada un dels sprints, i ho compararà amb el nombre esperat.

- Productivitat: Mesurarà el nombre d'activitats completades sobre el nombre d'activitats pertanyents a l'*sprint*.

- Costos directes: Mesurarà la diferència dels imports de costos directes amb els pressupostats inicialment.

- Costos indirectes i amortitzacions: Mesurarà la diferència dels imports dels costos indirectes i les amortitzacions del material usat.

Hem de tenir en compte, que els dos indicadors de costos, són directament dependents del de càrrega de treball. Ja que l'augment d'hores dedicades implica, de forma directa i proporcional, un augment en els costos de recursos humans.

Per a poder tenir una idea del valor d'una desviació, s'ha calculat el valor monetari que implica treballar una hora:

Tipus	Import (€)	
	CEO/CIO + Dev.	Desenvolupador
Costos Directes	28	8
Costos Indirectes	1,03	1,03
Amortitzacions	0,57	0,57
Total	29,60	9,60

Taula 20. Cost d'una hora extra

Tanmateix, hem de definir els valors dels indicadors. En quant aquests són en valors relatius, el seu rang és de 0 a infinit. Els valors òptims d'aquests estan definits en la següent taula:

Indicadors	Valor òptim
Càrrega de Treball	$\alpha = 1$
Productivitat	$\alpha = 1$
Costos directes	$\alpha <= 1$
Costos indirectes	$\alpha <= 1$

Taula 21. Valors òptims dels indicadors

Per a obtenir les α, es fan servir les següents formules:

Càrrega de treball	
El valor òptim s'aconsegueix en cas que les hores treballades siguin les mateixes que les planificades.	$\alpha = \frac{\#hores\ treballades}{\#hores\ planificades}$
Productivitat	
El valor òptim s'aconsegueix en cas que el nombre de tasques completades sigui igual al nombre de tasques planificades	$\alpha = \frac{\#tasques\ completades}{\#tasques\ planificades}$
Costos directes	
El valor òptim s'aconsegueix en cas que es treballin el mateix nombre d'hores planificades o menys	$\alpha = (\#hores\ treballades * S / costos\ directes)$ on S equival al salari corresponent
Costos indirectes i amortitzacions:	

El valor òptim s'aconsegueix en cas que es treballin el mateix nombre d'hores planificades o menys	$\alpha = (\#hores\ treballades * CIiApH) / CIiAP$ on CIiAPh són els costos indirectes i amortitzacions per hora, i CIiAP són els costos indirectes i amortitzacions pressupostats

Taula 22. Formules del control de gestió

En cas que qualsevol d'aquests indicadors mesurin una diferència del 10%, tant positivament com negativament, s'haurà de realitzar una reunió extraordinària per ajustar la planificació i recalcular el pressupost segons el nou calendari de terminis i entregues. L'objectiu és fer que aquests indicadors tornin a acostar-se als seus valors òptims.

8 Redisseny del sistema

Aquest capítol explica les modificacions en l'arquitectura del sistema i dels components que aquesta inclou.

8.1 Redisseny

8.1.1 Arquitectura

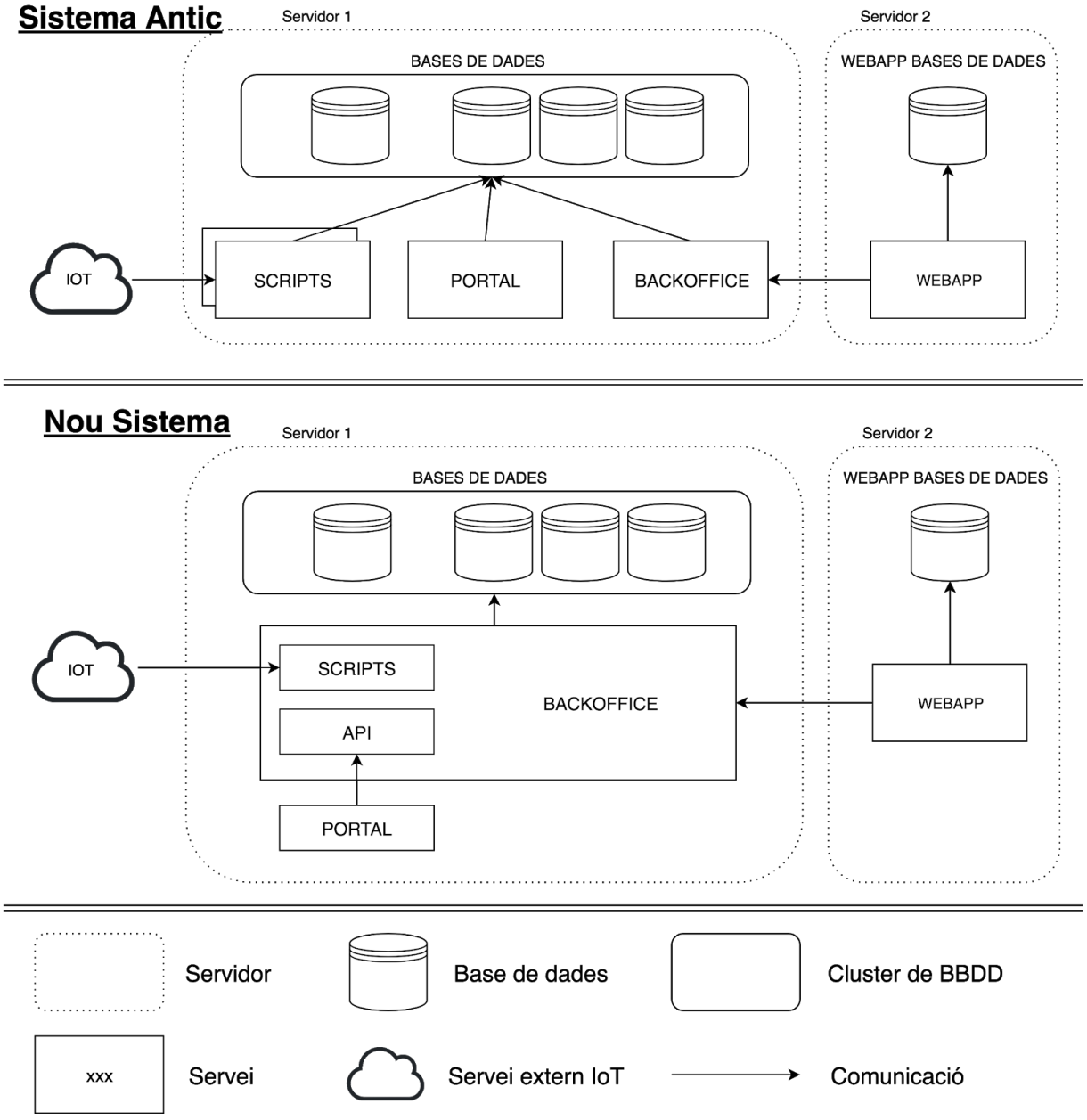


Figura 12. Comparació arquitectura

Per al que fa al servidor 1, amb el qual s'ha treballat durant la realització del projecte, els components finals de l'arquitectura són els marcats en l'esquema anterior.

El backoffice es consolida com a nucli de la nova arquitectura. L'aplicació web permet als membres de GreenCustomers poder administrar els elements del sistema i controlar l'estat dels hotels, sensors i nodes. El backoffice també inclou la lògica necessària per al descodificat i normalització de les dades provinents dels nodes instal·lats. I, absorbeix la lògica del back-end del portal per a re-aprofitar al màxim el codi. El backoffice conté una API REST on s'hi connectarà el portal per a demanar la informació requerida per l'usuari.

El portal seguirà sent la plataforma que utilitzaran els hotelers per a comprovar l'estat del seu hotel. Aquesta aplicació web però es converteix en una SPA (*Single Page Application*). És a dir, una aplicació amb només una pàgina HTML, on el seu contingut es modifica dinàmicament segons les accions de l'usuari [21]. A més, el *back-end* del portal, és substituït per l'API del backoffice.

Les bases de dades queden intactes, quant a l'arquitectura. La base de dades principal incorporarà tota la informació estructural i la resta de bases de dades se centraran en guardar les dades de consum de les habitacions d'hotel. Els canvis realitzats en l'estructura interna de cada base de dades es detallaran en el capítol "Noves Funcionalitats".

En canvi, en el servidor 2 no s'ha realitzat cap canvi, ni estructural ni en el comportament de la WebApp ni a l'estructura de la base de dades.

8.1.2 WebApp

El diagrama de la següent pàgina mostra els components que s'han fet servir i com es distribueixen segons la seva funció: model (ACTIVE RECORD), vista (ACTION VIEW) o controlador (ACTION CONTROLLER). En el cas de les vistes també s'ha afegit els components React que s'utilitza en cada una d'elles. Per entendre el diagrama, a continuació hi ha la llegenda:

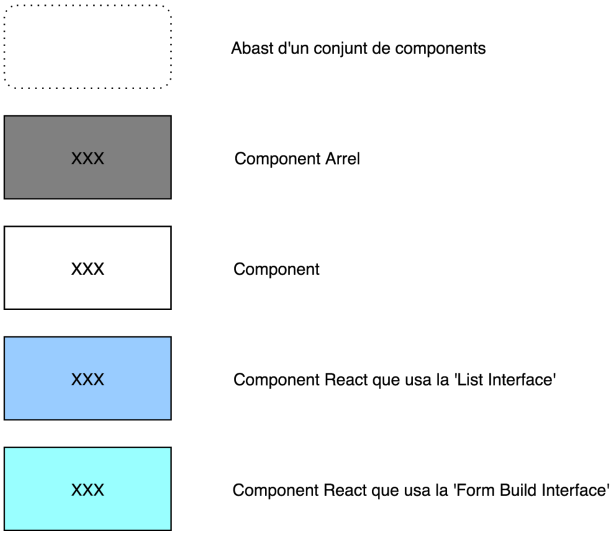


Figura 13. Llegenda diagrama de components de l'aplicació web

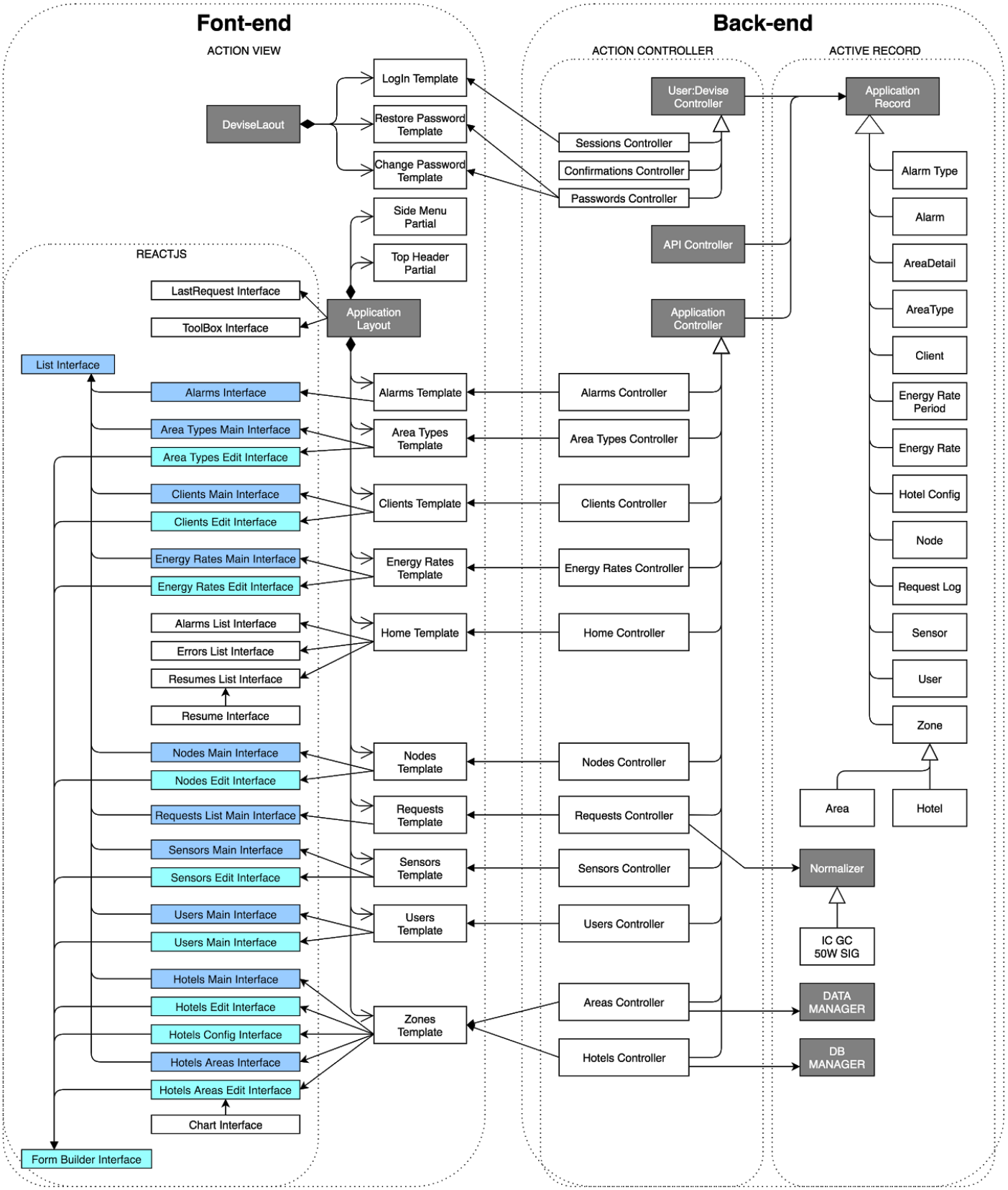


Figura 14. Diagrama de components de l'aplicació web

8.1.3 API

Dins de l'aplicació web, i tal com s'observa en el diagrama de components anterior, existeix un controlador anomenat api_controller. Aquesta classe s'encarrega de gestionar les peticions realitzades a l'API.

L'API és de tipus REST. Una API REST és un seguit de funcions que segueixen un conjunt de regles:

- URL base: Totes les operacions que ofereix se situen dins de l'abast d'un URL únic en l'aplicació. En el cas del projecte és *"/api"*.

- Mètodes HTTP estàndards: Els mètodes HTTP defineixen el caràcter de l'operació que es vol dur a terme. Els existents són: GET, POST, PUT, PATCH i DELTE. En l'API però només s'utilitzaran els mètodes GET i DELETE.

- Format: En una API REST, les peticions poden acceptar diferents formats de resposta. En aquesta versió, totes les respostes seran de format JSON, ja que aquesta API esta pensada per a ser usada des d'una aplicació Javascript.

- Codis de resposta: Quan es resol una petició, s'envia, a l'usuari que l'ha realitzat, la infomació demanada en cas que s'hagi resol correctament, o en cas d'error el missatge corresponent. Aquests missatges contenen un codi d'estat que aporta informació addicional del tipus d'error. Els codis usats en l'api són els següents:

CODI	DESCRIPCIÓ	EXPLICACIÓ
200	Èxit	L'operació s'ha realitzat correctament
401	No autoritzat	L'usuari no té una sessió oberta
404	Entitat no existent	L'entitat que s'ha buscat no existeix
412	Error de pre-condició	Falta algun dels paràmetres o, en cas que hi sigui, aquest no compleix el format esperat

Taula 23. Codis HTTP de l'API

Per a garantir la seguretat de les dades, totes les funcions, a excepció del "/login", estan protegides amb un token temporal que es genera a cada sessió amb una caducitat d'una hora a partir de l'última acció. Abans que l'acció demanda s'executi es valida el token enviat. En cas que el token no sigui valid o hagi caducat, l'acció que es vol dur a terme quedarà anul·lada i es notificarà a l'usuari amb el missatge d'error "User not logged in" i l'estatus 401.

METHOD	GET	URL	/login
DESCRIPCIÓ	Inicia la sessió a un usuari		
REQUEST			

paràmetres	{
	email : string,
	password : string
	}		

RESPONSE			

200	{
	email : string,
	active_token : string,
	forename : string,
	surname : string
	}		

401	{ error_message : “Wrong password” }		
404	{ error_message : “Wrong email” }		
412	{ error_message : “Email nor Password can’t be blank” }		
	{ error_message : “Email can’t be blank” }		
	{ error_message : “Password can’t be blank” }		

Taula 24. API /login

METHOD	DELETE	URL	/logout
DESCRIPCIÓ	Tanca la sessió a un usuari		
REQUEST			
paràmetres			
RESPONSE			
200	{}		

Taula 25. API /logout

METHOD	GET	URL	/hotel
DESCRIPCIÓ	Obté la informació del hotel el qual l'usuari pertany		
REQUEST			
paràmetres	-		
RESPONSE			
200	{ id : integer, name : string, subzones : { *Area* }, total_subzones : integer, config: *Config* }		
404	{ error_message : "Hotel not found" }		
DEFINICIONS			
Area	{ id : integer, name : string, subzones : { *Area* } *(opcional)* }		
Config	{ check_in : time, check_out : time, longitude : decimal, latitude : decimal, address : string, energy_rate : integer, p1 : decimal, p2 : decimal, p3 : decimal, p4 : decimal, p5 : decimal, p6 : decimal, rate : decimal, iee : decimal, e_vat : decimal, water_price : decimal, w_vat : decimal }		

Taula 26. API /hotel

METHOD	GET	URL	/hotel/:name
DESCRIPCIÓ	Obté la informació d'una àrea de l'hotel de l'usuari		
REQUEST			
paràmetres	-		
RESPONSE			
200	{ id : integer, name : string, type : string, details : *Detail*, components : [*Component*] }		
404	{ error_message : "Area not found" }		
DEFINICIONS			
Detail	{ objective_water : integer, objective_energy : integer, min_energy_day : integer, min_energy_hour : integer, eff_ap2 : decimal, eff_ap : decimal, eff_a : decimal, eff_b : decimal, eff_c : decimal, eff_d : decimal, eff_e : decimal, eff_f : decimal, }		
Component	{ node : string, number : integer, node_type : string, sensor : string, sensor_desc : string }		

Taula 27. API /hotel/name

METHOD	GET	URL	/hotel/:name/data/:from/:to/:tz
DESCRIPCIÓ	Obté els consums d'una àrea entre les dates "from" i "to" segons la zona horària "tz". Tant "from", "to" com "tz" no són obligatoris, i per defecte "to" és la data actual, "from" és la data de fa tres dies, i "tz" és la zona horària UTC (GMT +0).		
REQUEST			
paràmetres	-		
RESPONSE			
200	{ data : [*Data*], data_sub : [*Data*], from : integer, to : integer, tz : integer }		
404	{ error_message : "Area not found" }		
412	{ error_message : "Wrong "from" date format. Use YYYY-MM-DD " }		
	{ error_message : "Wrong "to" date format. Use YYYY-MM-DD " }		
	{ error_message : "Wrong "time zone" format. Use +00:00 or -00:00 " }		
DEFINICIONS			
Data	{ date_time : string, energy : decimal, cold_water : decimal, hot_water : decimal, co2 : decimal, min_e_hour : decimal, }		

Taula 28. API /hotel/name/data

8.2 Implementació

En aquest apartat s'explicara com s'ha dut a terme el redisseny de l'arquitectura: quines tecnologies s'han usat, com s'estructura el codi, i com s'estructuren els components que conformen el sistema.

8.2.1 Tecnologies usades

Entorn de desenvolupament i de proves

En l'entorn de desenvolupament s'ha usat un contenidor Docker. Docker és una eina que permet encapsular un projecte/aplicació en un contenidor que defineix un entorn segur i compatible amb qualsevol sistema [22][23]. Un contenidor genera una màquina virtual pròpia per a executar l'aplicació que conté segons els paràmetres definits. D'aquesta manera s'aconsegueix que l'aplicació, independentment de les tecnologies que utilitzi i les seves versions, sigui completament compatible amb el sistema operatiu de la màquina física on es vol executar.

Per a l'entorn de proves s'utilitza un VPN en un proveïdor extern on s'ha instal·lat el sistema operatiu basat en UNIX Debian9 Strech. Debian és un sistema operatiu opensource desenvolupat en el marc de "The Debian Project" [24]. Sobre el sistema operatiu s'utilitza el servei NginX amb PushionPassenger per a executar un servidor HTTP web.

Aplicació Web

Per al desenvolupament del *back-end* de l'aplicació web s'ha utilitzat el llenguatge de programació *ruby* 2.5.0 amb el *framework* Ruby On Rails 5.2.1 (RoR). RoR és un *framework* basat en el patró Model-Vista-Controlador i ActiveRecord (Explicació en detall en el punt 8.2.3). Pel front-end, a més de les tres tecnologies base per al desenvolupament de pàgines web (HTML5, CSS3 i Javascript ECMAScript 2018), s'ha fet ús de les llibreries ReactJS, que facilita la programació de UIs (User Interfaces) amb l'ús de components reactius a les accions dels usuaris, i Bable, que tradueix la sintaxi JSX (Javascript XML) que utilitza ReactJS en sintaxi compatible segons l'especificació de l'ECMAScript 5.

Bases de dades

La informació s'emmagatzema en bases de dades relacionals amb tecnologia MySQL v5.7.17 amb el motor InnoDB. Per a l'accés a les dades i la seva modificació s'usa el llenguatge declaratiu SQL [25].

API

L'API desenvolupada és de tipus REST. Això vol dir que la interfície ofereix només un conjunt d'operacions fixes que funcionen amb el protocol HTTP (i HTTPS) [26]. Per a la transferència

de dades en les operacions, s'ha utilitzat el format estàndard JSON (Javascript Object Notation) [27].

SMTP

L'aplicació utilitza SMTP (Simple Mail Transfer Protocol) per a comunicar a un usuari o client que se'ls ha permès l'accés al backoffice o al portal respectivament [28]. El servei que s'utilitza, i només operatiu en l'entorn de proves, és el propi proveït pel VPN on resideix el servidor. Per a realitzar l'enviament dels correus, RoR conté un mòdul que funciona d'interfície interna i s'encarrega de comunicar-se amb el servei triat en la configuració del mòdul [29].

8.2.2 Estructura del codi

El codi segueix l'estructura bàsica d'una aplicació RoR. Des de la carpeta arrel del projecte hi trobem les següents sub-carpetes:

/app	Conté tot el contingut de l'aplicació en si
/assets	
/images	Conjunt d'imatges que es mostren
/javascripts	Conjunt de fitxers Javascript que s'executaran al client
/components	Conjunt de components de ReactJS
/stylesheets	Conjunt de fitxers d'estils que s'aplicaran a les vistes
/controllers	Classes que actuen de controladors en el patró MVC
/helpers	Conjunt de mòduls que simplifiquen la lògica de renderització de les vistes
/mailers	Conjunt de mòduls per enviar correus electrònics via SMTP
/models	Conjunt de classes que actuen de models en el patró MVC
/views	Conjunt de classes que actuen de vistes en el patró MVC
/config	Carpeta que conté els fitxers de configuració de l'aplicació
/db	Carpeta amb el conjunt de fitxer de migracions i l'esquema de les bbdd
/log	Carpeta amb els fitxers dels logs
/public	Capeta amb el conjunt de fitxers públics de l'aplicació
/test	Carpeta amb el conjunt de fitxers usats en els tests
/tmp	Carpeta amb els fitxers temporals generats en temp d'execució

Taula 29. Estructura del codi

8.2.3 Implementació de la WebApp

L'aplicació segueix el patró de disseny MVC. En RoR els models fan servir *ActiveRecord*, les vistes *ActionView* i els controladors *ActionController*.

ActiveRecord és un patró d'arquitectura que usa interfícies per a la comunicació d'una aplicació web amb una base de dades. Les funcions són de caràcter CRUD (*Create, Read, Update, Delete*) [30] [31]. A més utilitza l'*ORM Framework* (Object Relational Mapping). L'ORM permet, entre altres, representar les associacions entre models, herència de classes i validació dels models abans de confirmar els canvis en la base de dades. En RoR, existeix la interfície *ApplicationRecord* que és una extensió d'ActiveRecord::Base (la classe base d'ActiveRecord). *ApplicationRecord* és la classe principal de tots els models de l'aplicació per defecte.

ActionController és la classe primària de tots els controladors de l'aplicació. Quan es realitza una petició a l'aplicació, primer es resol quina acció s'ha d'executar mitjançant el router i després s'executa la funció corresponent. Cada funció del controlador té accés als paràmetres de la petició i al final de l'acció ha de generar un output. Per defecte aquest output és una vista, en la qual hi pot enviar paràmetres i/o variables. Però també es pot respondre l'acció amb data en JSON, XML o CSV per exemple. *ActionController* té accés als models de l'aplicacció, per aquest motiu actua com a intermediari entre les vistes i els models [32].

En RoR, les vistes, creades a partir d'HTML, poden incorporar l'extensió ERB (*Embedded Ruby*). Aquest fitxer admet codi ruby que s'inclogui dins de tags amb una sintaxis específica. *ActionView* s'encarrega de compilar i construir l'html final que el navegador web mostrarà. Un fitxer HTML-ERB pot tenir tres funcions: plantilla (*template*), "parcial" (*partial*) o pla (*layout*) [33]:

- Els templates són fitxers HTML-ERB que mostren un contingut específic. Amb l'ERB i els paràmetres de resposta que envia el controlador es poden obtenir diferents resultats HTML de la compilació del mateix fitxer.

- Els *layouts* són la peça clau d'*ActionView*, en el *layout* es dissenya l'aspecte general de la interfície d'usuari. En aquest disseny s'ha de mantenir un espai on s'hi voldrà mostrar el contingut principal de la pàgina. En aquest espai s'hi renderitzarà el *template*. Amb aquesta tàctica, diferents pàgines poden re-utilitzar un mateix disseny i només canviar el cos principal de la pàgina.

- Els *partials* són fitxers que s'utilitzen per encapsular codi HTML-ERB. Un *partial* es pot renderitzar dins de qualsevol vista, ja sigui *template*, *layout* o un altre *partial*. S'utilitzen per mantenir el codi net quan un contingut s'ha de repetir múltiples vegades.

8.2.5 Implementació SMTP

RoR utilitza el mòdul *ActionMailer* per a la gestió de l'enviament dels correus via SMTP [29]. Inicialment s'ha de configurar el servei en el fitxer d'entorn de proves on s'ha d'especificar quin proveïdor de correu s'utilitza, l'usuari i la contrasenya.

ActionMailer és una classe accessible des de qualsevol controlador. Aquest conté funcions, les quals cada una s'encarrega d'enviar un tipus d'email. En el nostre cas, n'hi ha dues: una per enviar el correu de benvinguda a un nou usuari del backoffice, i un altre per donar la benvinguda a un nou client. Per a construir el cos del correu electrònic, a més d'acceptar text sense format, ActionMailer admet l'ús d'ActionView amb totes les seves funcionalitats. Els correus es basen sobre un *layout* i el seu contingut es construeix a partir d'un *template*.

9. Redisseny de la interfície

Les interfícies s'han re-dissenyat per a que siguin més netes i intuïtives de cara a l'usuari.

9.1 *Layout* de Devise

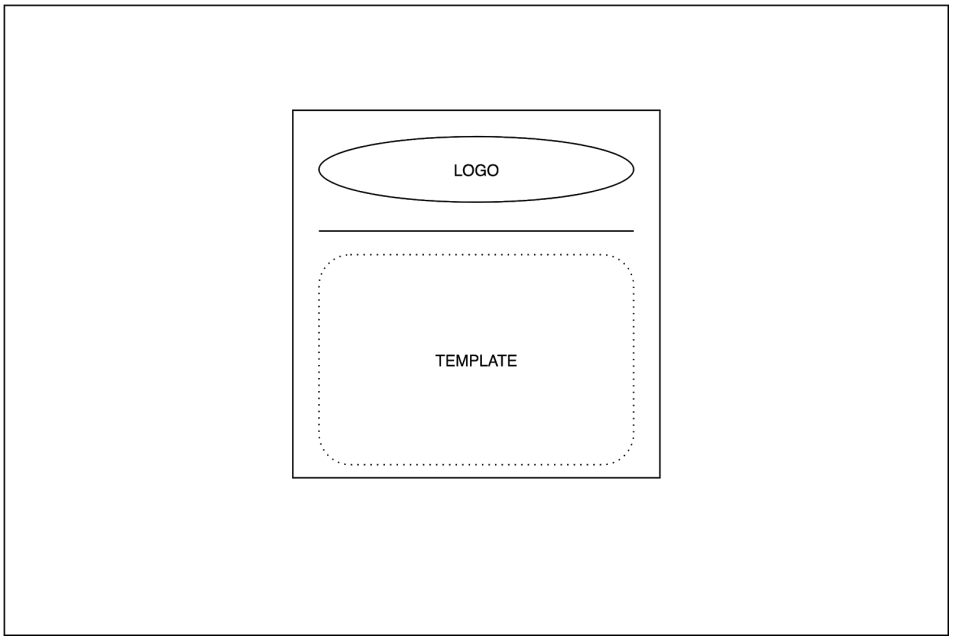


Figura 15. MockUp del Layout de Devise

El layout es distribueix amb dos elements: el fons que és un degradat subtil entre dues tonalitats de verd (#89c400), i l'element central de fons blanc, on s'hi renderitza el template, el contingut de la vista corresponent. En total hi ha tres templates, un per a cada vista: el *login*, el *restore password*, i el *change password*:

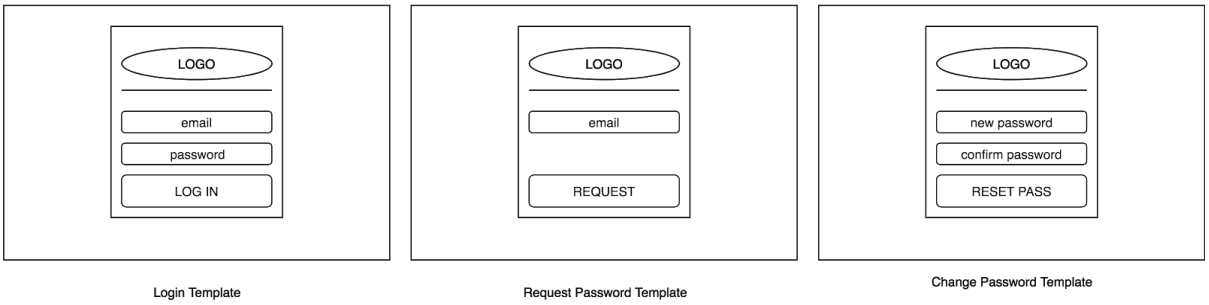


Figura 16. Mockups dels templates de Devise

9.2 *Application Layout*

L'*Application Layout* és el layout principal que utilitzen la majoria de les vistes. Aquesta es distribueix en Left-Main en l'eix horitzontal i en Header-Content en el vertical:

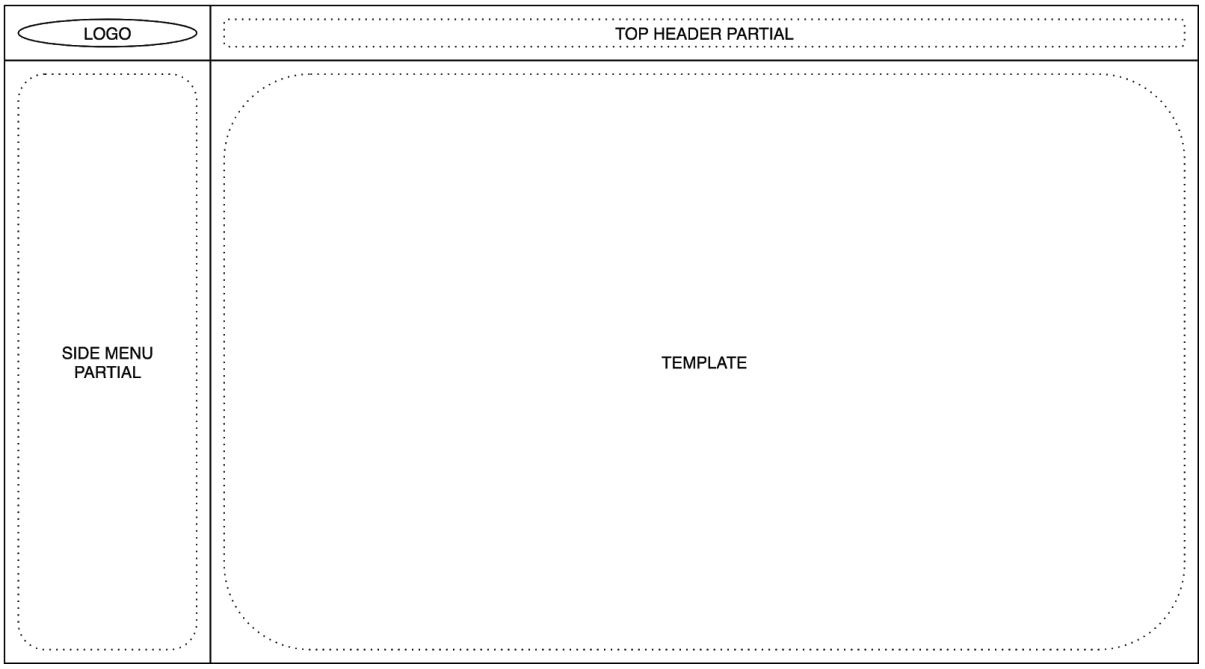


Figura 17. Application Layout

Amb el side menu i el top header, la vista del layout queda amb la següent distribució:

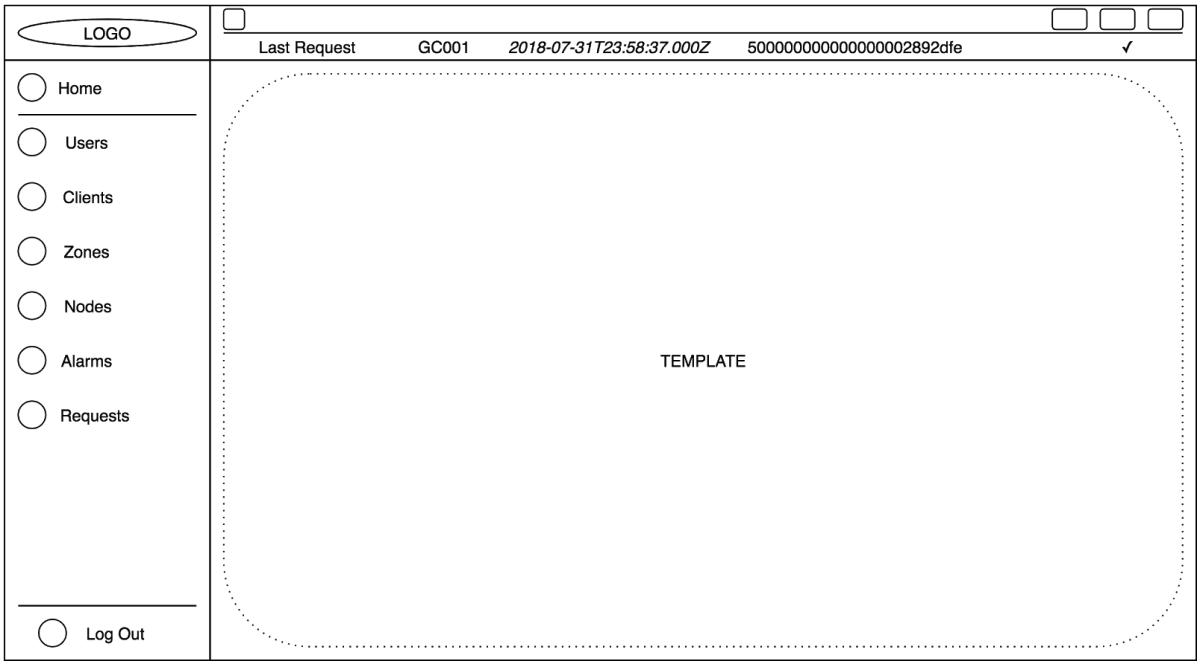


Figura 18. Application Layout amb els partials

El side menu és el menú de navegació de la pàgina. Cada element, a excepció de “Zones” i “Nodes”, genera una navegació a la vista que anuncia. Els elements de “Zones” i “Nodes” són menús desplegables amb tres possibles opcions cada un. El sub-menú de “Zones” té els links a: “Hotels”, “Energy Rates” i “Area Types”. I el sub-menú de “Nodes” té els links a: “All Nodes”, “Node Types” i “Sensors”.

Pel que fa als templates, la distribució entre ells és semblant i es s’agrupen en tres tipus:

9.2.1 Home Template

La vista de “Home” té un templete amb una distribució única. El marc del template està dividit verticalment en dos: la zona esquerra i principal hi ha un llistat de gràfics amb els consums de cada hotel de les últimes 24 hores. En la part dreta, hi ha dos llistats amb les últimes alarmes que s’han generat i amb les últimes requests que han generat algún error previst:

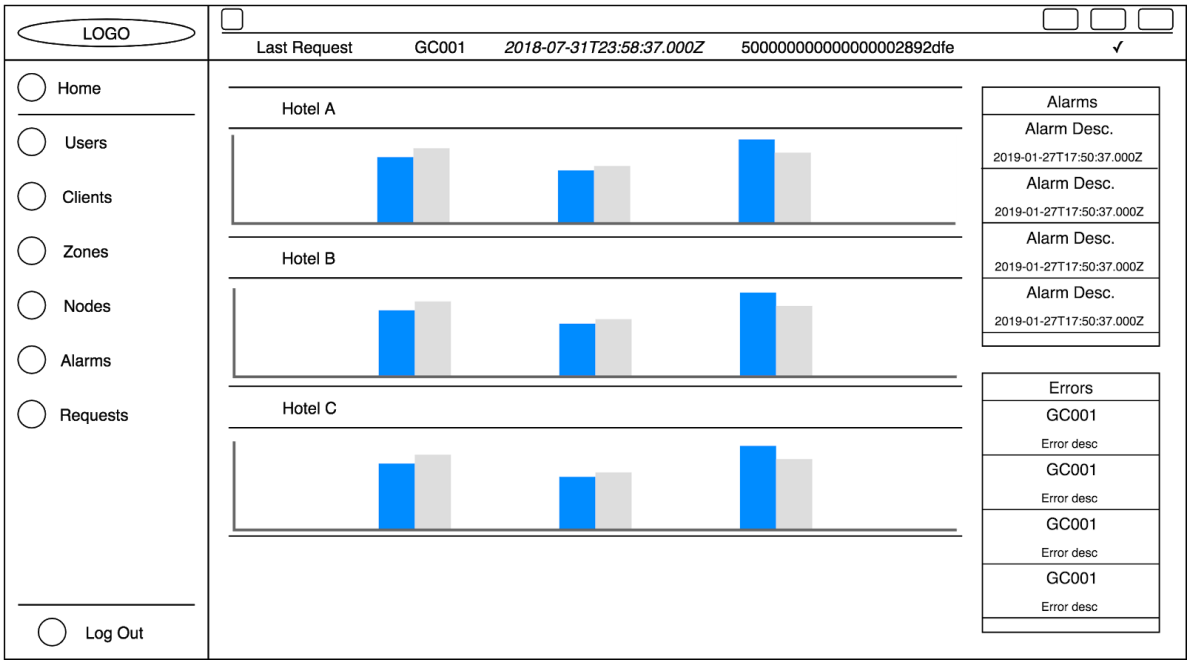


Figura 19. Mockup del home template

9.2.2 Alarms i Requests Templates

Aquestes dues templates segueixen un segon model de distribució dels elements. Són resumidament vistes que únicament contenen una llista amb informació sobre les alarmes i les requests rebudes en el sistema respectivament.

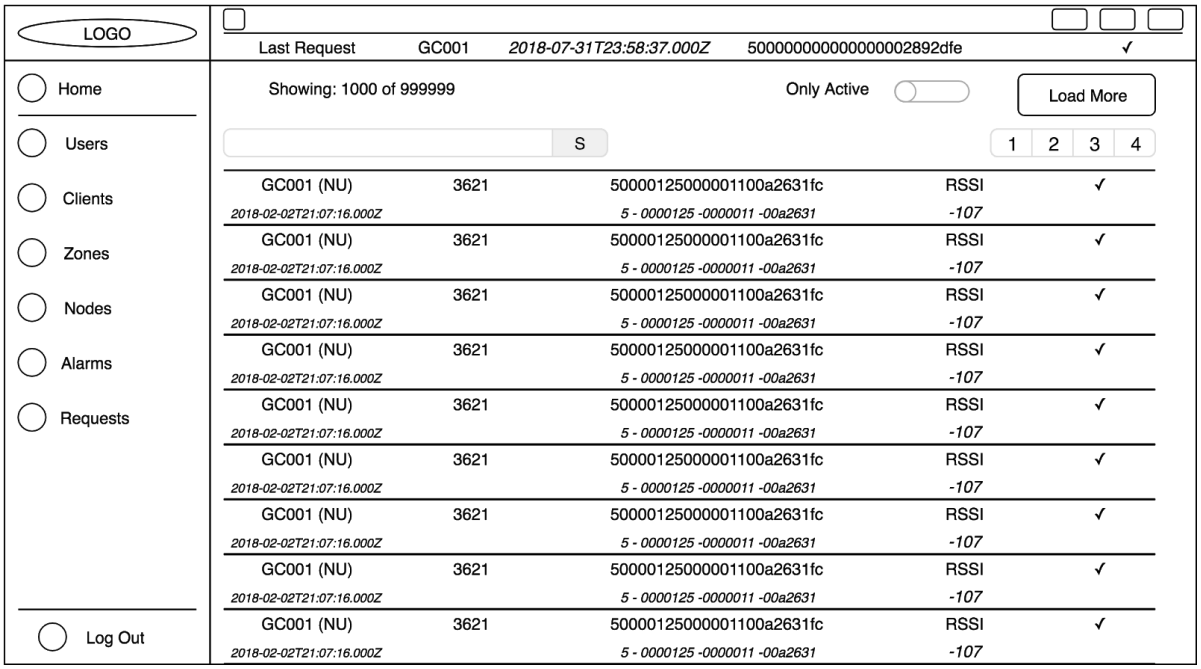


Figura 20. Mockup Alarms i Errors Template

9.2.3 Default Template

Per a la resta d'interfícies, la distribució dels elements en el template és la mateixa. A la dreta se siuta el llistat d'elements de la interfície, i a l'esquerra el contingut canvia dinàmicament: o es mostra el formulari per a crear una entitat nova o el formulari per editar-ne una d'existent. En algunes de les interfícies, el formulari d'edició està separat en pestanyes per a facilitar-ne la lectura i la cerca dels camps.

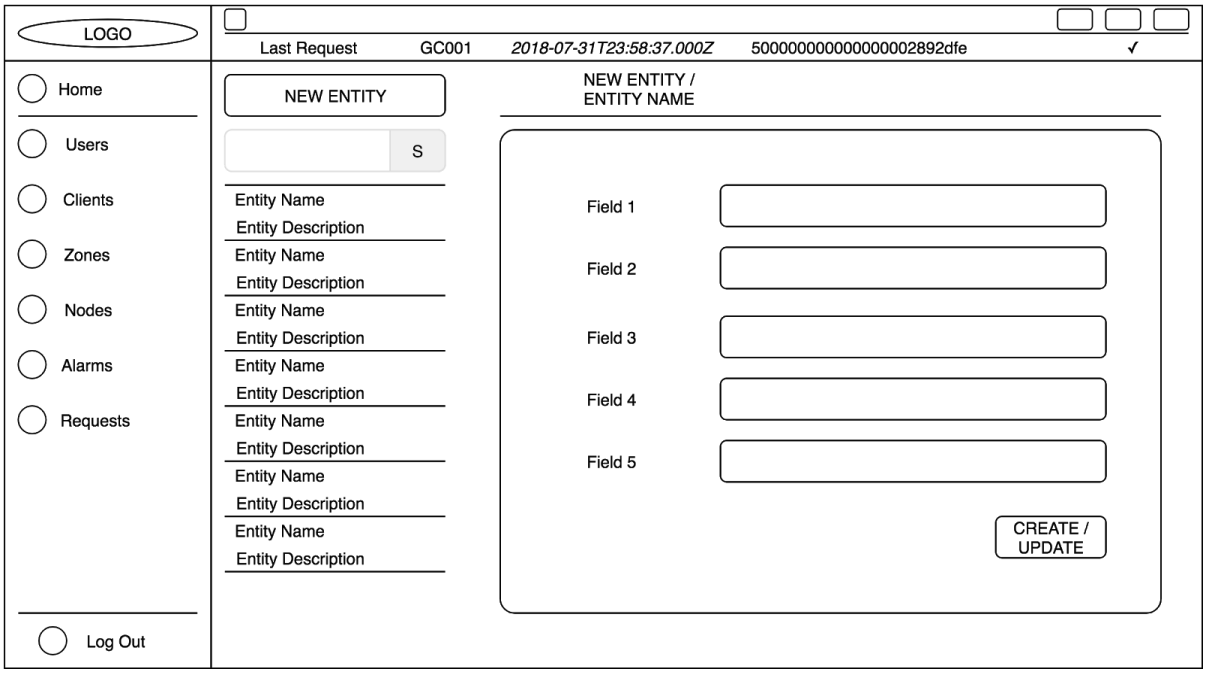


Figura 21. Mockup default template

10. Noves funcionalitats

A més de re-dissenyar el sistema, aquest projecte incorpora noves funcionalitats que n'augmenten els possibles usos de l'aplicació. En aquest capítol es descriurà l'especificació i el disseny de les noves funcionalitats incorporades. No només les determinades en l'anàlisi de problemes i debilitats realitzat en el capítol 3, sinó també les que s'han proposat durant la realització del projecte.

10.1 Especificació

En l'especificació es determinaran els casos d'ús per a cada nova funcionalitat i quins requisits s'han de complir per a l'execució satisfactòria de cada un. Els nous requisits són de caràcter funcional, ja que les noves funcionalitats hauran de mantenir els requisits no funcionals aplicats en la fase de redisseny del sistema. Les funcionalitats inclouran mètodes per a crear, editar i llegir entitats, però també eliminar-ne. En aquest últim cas, el procés d'eliminació es durà a terme en dos passos seguint el mètode *soft-delete*. Aquest mètode consta a no esborrar completament una entitat del sistema, sinó que tracte d'activar un *flag* que marqui l'entitat com a desactivada/eliminada. L'entitat serà visible des del seu panell, però no podrà interactuar amb d'altres. Un cop una entitat ha sigut marcada com a eliminada, es podrà optar a esborrar-la completament.

10.1.1 Tipus de node

En la debilitat 3 (Explicada en el punt 3.2.3 del document), es determina la necessitat d'incorporar els tipus de nodes presents en el sistema. La incorporació dels tipus de node com a model de dades en el sistema implica la creació del model en si mateix el qual conté els attributs:

- id: Identificador del tipus de node.
- *'brand'*: marca del proveïdor del node.
- *'model'*: model del node.
- *'location'*: zona geogràfica per a distingir la freqüència que utilitza.
- *'normalizer'*: codi per a saber el procés que s'ha d'utilitzar per a descodificar les dades que envia.

Sobre aquests atributs hi regeixen quatre requisits:

- L'*id* del tipus de node ha de ser únic.
- El nom de la marca, representat per l'atribut *brand* ha de ser únic.
- Per a una marca, no pot haver-hi dos *models* amb el mateix nom.
- Tots els camps són obligatoris.

De la nova classe en deriven sis accions per a permetre la gestió dels tipus de node existents en el sistema:

- Llegir el llistat de tipus de node
- Llegir la informació d'un tipus de node
- Alta d'un nou tipus de node en el sistema
- Modificació d'un registre de tipus de node
- Eliminar un registre de tipus de node
- Assignació d'un node al tipus el que pertany

Per a poder efectuar la última acció s'ha hagut de modificar la classe *Node*. Més concretament s'hi ha afegit l'attribut:

- *node_type_id*: Identificador del *node_type* el qual el node està assignat.

Aquest atribut és obligatori, no pot tenir valor nul.

Per tant, amb el llistat dels atributs, el model queda representat per la classe *NodeType* i, les accions sobre aquesta es modelitzen amb el diagrama de casos d'ús que hi ha a continuació:

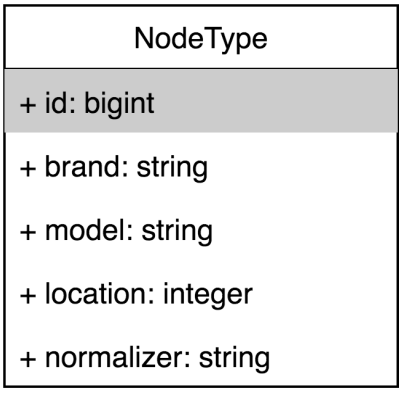


Figura 22. Especificació classe NodeType

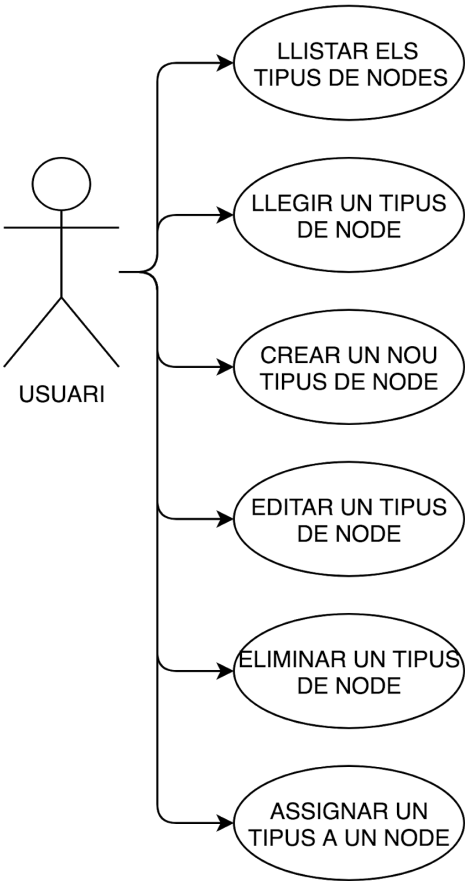


Figura 23. Casos d'ús de NodeType

En les taules de sota, es descriuen els requisits dels casos d'usos que afegeix la nova funcionalitat:

Cas d'ús:	Llistar els tipus de node	Actor:	Usuari
Precondició:	-		
Disparador:	L'usuari vol obtenir un llistat amb tots els tipus de node		
Escenari d'èxti			
1. L'usuari navega a la pàgina Node Types dins del sub-menú Nodes del menú esquerra 2. El sistema mostra el llistat de tipus de nodes disponibles			
Extensions:			
- 1a L'usuari cancel·la l'acció			

Cas d'ús:	Llegir un tipus de node	Actor:	Usuari
Precondició:	Hi ha almenys un tipus de node creat		
Disparador:	L'usuari vol llegir la informació d'un tipus de node		
Escenari d'èxti			
1. L'usuari navega a la pàgina Node Types dins del sub-menú Nodes del menú esquerra 2. El sistema mostra el llistat de tipus de nodes disponibles 3. L'usuari selecciona un element de la llista 4. El sistema carrega les dades del tipus de node seleccionat			
Extensions:			
- 1a, 3a L'usuari cancel·la l'acció			

Cas d'ús:	Crear nou Tipus de Node	Actor:	Usuari
Precondició:	-		
Disparador:	L'usuari vol crear un nou tipus de node		
Escenari d'èxti			
1. L'usuari omple els camps de brand, model, location i normalizer 2. L'usuari confirma el formulari 3. El sistema comprova les dades introduïdes 4. El sistema registre el nou tipus de node 5. El sistema notifica l'èxit de la operació			
Extensions:			
- 1a L'usuari cancel·la l'acció - 3a Les dades introduïdes no són vàlides			

- 3a1 Es mostra un missatge d'error i es retorna al punt 1

Cas d'ús:	Editar un Tipus de Node	Actor:	Usuari
Precondició:	Hi ha almenys un tipus de node creat		
Disparador:	L'usuari vol editar un tipus de node		

Escenari d'èxti

1. L'usuari llegeix la informació d'un tipus de node
2. L'usuari modifica algun dels camps: brand, model, location i normalizer
3. L'usuari confirma el formulari
4. El sistema comprova les dades introduïdes
5. El sistema modifica el tipus de node
6. El sistema notifica l'èxit de la operació

Extensions:

- 1a,2a L'usuari cancel·la l'acció
- 4a Les dades introduïdes no són vàlides
    - 4a1 Es mostra un missatge d'error i es retorna al punt 2

Cas d'ús:	Eliminar un Tipus de Node	Actor:	Usuari
Precondició:	Hi ha almenys un tipus de node creat		
Disparador:	L'usuari vol eliminar un tipus de node		

Escenari d'èxti

1. L'usuari llegeix la informació d'un tipus de node
2. L'usuari fa click al botó d'eliminar
3. L'usuari confirma l'acció
4. El sistema comprova la validesa de la operació
5. El sistema modifica el tipus de node
6. El sistema notifica l'èxit de l'operació

Extensions:

- 1a,2a L'usuari cancel·la l'acció
- 4a L'operació no es pot dur a terme
    - 4a1 Es mostra un missatge d'error i es retorna al punt 2

Cas d'ús:	Assignar un Tipus a un Node	Actor:	Usuari
Precondició:	Hi ha almenys un node i un tipus de node creat		
Disparador:	L'usuari vol assignar un tipus a un node		
Escenari d'èxti			
1. L'usuari llegeix la informació d'un node 2. L'usuari modifica el nou camp: tipus de node 3. L'usuari confirma el formulari 4. El sistema comprova les dades introduïdes 5. El sistema modifica el node 6. El sistema notifica l'èxit de la operació			
Extensions:			
- 1a,2a L'usuari cancel·la l'acció - 4a Les dades introduïdes no són vàlides - 4a1 Es mostra un missatge d'error i es retorna al punt 2			

10.1.2 Jerarquització de les àrees

En la debilitat 4 (Explicada en el punt 3.2.4 del document), es detalla per què el nou sistema ha d'incorporar la possibilitat de jerarquitzar les àrees. La jerarquització s'ha de realitzar permetent un nombre indefinit de nivells: la relació *Area-Subareas* ha de ser recursiva sobre el mateix model *Area*. El sistema actual només comprèn el model *Hotel* que té múltiples àrees. La nova relació ha de seguir el mateix patró entre dues instàncies d'*Area*. Aquesta relació té però 2 requisits que mantenen un estat vàlid de totes les instàncies:

- Dues àrees només poden estar jerarquitzades entre si, independentment del nombre de nivells intermedis, si les dues àrees pertanyen al mateix hotel.

- La jerarquització de les àrees no pot crear cicles: Una àrea A no pot categoritzar-se com a sub-àrea d'una àrea B, si aquesta és, independement del nombre de nivells intermedis, sub-àrea d'A.

La funcionalitat la recull un sol cas d'ús amb la seva taula del requisit funcional:

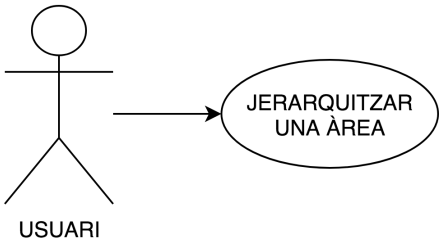


Figura 24. Cas d'ús Jerarquitzar una àrea

Cas d'ús:	Jerarquitzar una àrea	Actor:	Usuari
Precondició:	Hi ha almenys dues àrees en un hotel		
Disparador:	L'usuari vol jerarquitzar una àrea		
Escenari d'èxti			
1. L'usuari llegeix la informació d'una àrea 2. L'usuari selecciona l'àrea 'pare' 3. L'usuari confirma el formulari 4. El sistema comprova les dades introduïdes 5. El sistema modifica l'àrea 6. El sistema notifica l'èxit de la operació			
Extensions:			
- 1a,2a L'usuari cancel·la l'acció - 4a Les dades introduïdes no són vàlides: es crea un cicle o les àrees no pertanyen al mateix hotel - 4a1 Es mostra un missatge d'error i es retorna al punt 2			

10.1.3 Indeterminar el nombre sensors per node

El nou sistema, com estableix la debilitat 5 en el punt 3.2.5, ha d'incorporar la capacitat de crear nodes sense restriccions sobre el número de sensors els quals s'hi puguin connectar. Cada connexió que un node pot tenir és representada pel model *Component*. Les instàncies de la classe estan associades a un Node, i poden tenir associat, de forma optional un *Sensor* i una *Area*. El nombre de components que té un node, depèn directament del seu tipus. Per a mantenir una relació estable entre *NodeType* i *Node*, i poder saber les característiques de cada connexió, s'afegirà el model *Interface*, que virtualitza les interfícies físiques que cada tipus de node té en realitat. La relació que mantenen *NodeType* i *Node* de forma directa es manté en les clases Interface i Component de forma implícita sense haver de crear cap associació entre les classes. D'aquesta nova funcionalitat en derives dos models nous:

10.1.3.1 Creació del model *Interface*

El model *Interface* és la representació dels punts de connexió real que té un tipus de node. En termes tècnics, els ports I/O de la placa del node. Cada port s'identifica per el seu número, a més del número de serie del node. Cada pin admet una connexió d'un tipus concret: digital, analògica o de I2C, i quina direcció té la connexió: si és d'entrada o de sortida. Per tant, una instància d'Interface emmagatzemarà, a més de l'identificador del tipus de node al qual pertany, el número del pin (el qual em referirem com a canal o *channel*), el tipus de connexió i el sentit de la connexió.

Per a mantenir un estat correcte sobre el número de cada *Interface* d'un node, aquest s'actualitza automàticament segons les interfícies resultats de les operacions de creació i eliminació. Quan es crea una interfície, s'hi assigna el número corresponent, i quan s'elimina una, la resta s'han d'actualitzar per a mantenir una enumeració constant. El comportament del número les següents restriccions:

- Quan s'afegeix una interfície en un tipus de node, el número assignat ha de ser igual al nombre existent d'interficies prèvies a la creació d'aquest més 1.

- Quan s'elimina una interfície d'un tipus de node, s'han d'actualitzar els números de les interfícies per a que concordin.

Finalment, només es permetrà modificar (crear, editar i/o eliminar) interfícies quan no hi hagi cap node del tipus actiu al sistema.

De tot, les accions que en deriven i que podrà dur a terme l'usuari sobre el nou model són:

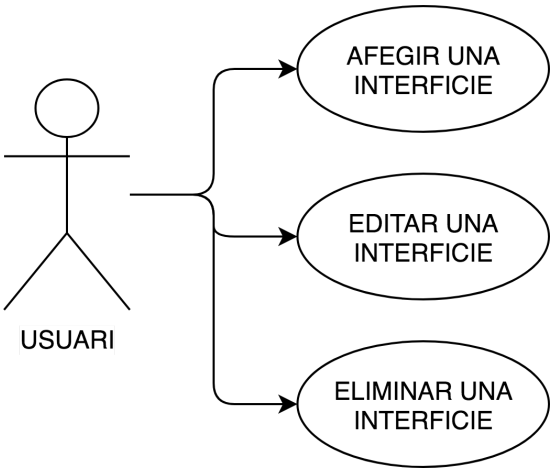


Figura 25. Casos d'ús de NodeType Interfaces

I els seus requisits funcionals associats són:

Cas d'ús:	Afegir una interfície	Actor:	Usuari
Precondició:	Hi ha almenys un tipus de node		
Disparador:	L'usuari vol afegir una interfície en un tipus de node		
Escenari d'èxti			
1. L'usuari llegeix la informació d'una tipus de node 2. L'usuari omple el formulari de nova interfície 3. L'usuari confirma el formulari 4. El sistema comprova les dades introduïdes 5. El sistema crea una nova interfície 6. El sistema notifica l'èxit de la operació			
Extensions:			
- 1a,2a L'usuari cancel·la l'acció			

Cas d'ús:	Editar una interfície	Actor:	Usuari
Precondició:	Hi ha un tipus de node amb almenys una interfície		
Disparador:	L'usuari vol editar una interfície		
Escenari d'èxti			
1. L'usuari llegeix la informació d'una tipus de node 2. L'usuari modifica la informació d'una interfície 3. L'usuari confirma el formulari 4. El sistema comprova les dades introduïdes 5. El sistema modifica la interfície 6. El sistema notifica l'èxit de la operació			
Extensions:			
- 1a,2a L'usuari cancel·la l'acció - 4a Les dades introduïdes no són vàlides: hi ha nodes associats amb sensors connectats a la interfície - 4a1 Es mostra un missatge d'error - 4a2 Es retorna al punt 2			

Cas d'ús:	Eliminar una interfície	Actor:	Usuari
Precondició:	Hi ha un tipus de node amb almenys una interfície		
Disparador:	L'usuari vol eliminar una interfície		
Escenari d'èxti			
1. L'usuari llegeix la informació d'una tipus de node 2. L'usuari clica el botó d'eliminar interfície 3. L'usuari confirma el formulari 4. El sistema comprova les dades introduïdes 5. El sistema elimina la interfície 6. El sistema notifica l'èxit de la operació			
Extensions:			
- 1a,2a L'usuari cancel·la l'acció - 4a Les dades introduïdes no són vàlides: hi ha nodes associats amb sensors connectats a la interfície - 4a1 Es mostra un missatge d'error - 4a2 Es retorna al punt 2			

El nou model Interface amb la relació amb la classe NodeType, queda recollit en següent fragment del diagrama de classes:

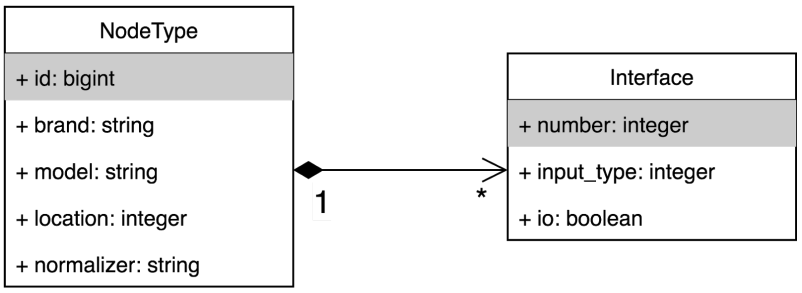


Figura 26. Especificació classe Interface

10.1.3.2 Creació del model *Component*

Un component és la representació de cada port d'un node i, com ja s'ha especificat, se li pot connectar un sensor, i assignar-ne una àrea. Un component manté una relació directe amb les interfícies del tipus de node associat. Per tant, aquest només es crea o s'elimina si es crea o s'elimina una *Interface* del tipus de node associat. D'aquesta manera, les úniques operacions disponibles que pot dur a terme l'usuari sobre un component són les de connectar/desconnectar un sensor i assignar una àrea. En la primera acció s'hi ha de tenir en compte la restricció implícita que representa l'acció de connectar un sensor:

- Només es pot connectar un sensor a un component si aquest representa una interfície de tipus *Digital* i és un port d'entrada.

Les accions que un usuari pot dur a terme respecte una instancia de la classe Component són:

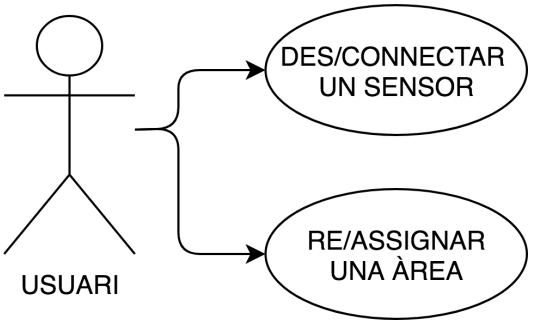


Figura 27. Casos d'ús de Components

I les requisits funcionals respectius:

Cas d'ús:	Connectar un sensor	Actor:	Usuari
Precondició:	Hi ha un node amb un component i un sensor		
Disparador:	L'usuari vol connectar un sensor		
Escenari d'èxti			
1. L'usuari llegeix la informació d'un node 2. L'usuari selecciona el sensor que vol conectar del camp del formulari d'edició 3. L'usuari confirma el formulari 4. El sistema comprova les dades introduïdes 5. El sistema actualitza el component 6. El sistema notifica l'èxit de la operació			
Extensions:			

- 1a,2a L'usuari cancel·la l'acció
- 4a El sensor no és compatible amb la interfície
  - 4a1 Es mostra un missatge d'error
  - 4a2 Es retorna al punt 2			

Cas d'ús:	Desconnectar un sensor	Actor:	Usuari
Precondició:	Hi ha un node amb un component amb un sensor connectat		
Disparador:	L'usuari vol desconnectar un sensor		
Escenari d'èxti			
1. L'usuari llegeix la informació d'un node 2. L'usuari desselecciona el sensor del camp del formulari d'edició 3. L'usuari confirma el formulari 4. El sistema comprova les dades introduïdes 5. El sistema actualitza el component 6. El sistema notifica l'èxit de la operació			
Extensions:			
- 1a,2a L'usuari cancel·la l'acció			

Cas d'ús:	Assignar una àrea	Actor:	Usuari
Precondició:	Hi ha un node amb un component, i una àrea		
Disparador:	L'usuari vol assignar una àrea		
Escenari d'èxti			
1. L'usuari llegeix la informació d'un node 2. L'usuari selecciona l'àrea que vol assignar del camp del formulari d'edició 3. L'usuari confirma el formulari 4. El sistema comprova les dades introduïdes 5. El sistema actualitza el component 6. El sistema notifica l'èxit de la operació			
Extensions:			
- 1a,2a L'usuari cancel·la l'acció			

Cas d'ús:	Des/Re-Assignar una àrea	Actor:	Usuari
Precondició:	Hi ha un node amb un component, i una àrea		
Disparador:	L'usuari vol des-assignar o re-assignar una àrea		
Escenari d'èxti			
1. L'usuari llegeix la informació d'un node 2. L'usuari selecciona l'àrea que vol assignar del camp del formulari d'edició, o el deixarà vuit 3. L'usuari confirma el formulari 4. El sistema comprova les dades introduïdes 5. El sistema actualitza el component 6. El sistema notifica l'èxit de la operació			
Extensions:			
- 1a,2a L'usuari cancel·la l'acció			

En resum, les associacions entre les classes Node, NodeType, Sensor, Area i les noves classes Interface i Component queden descrites en el fragment de diagrama de classes següent:

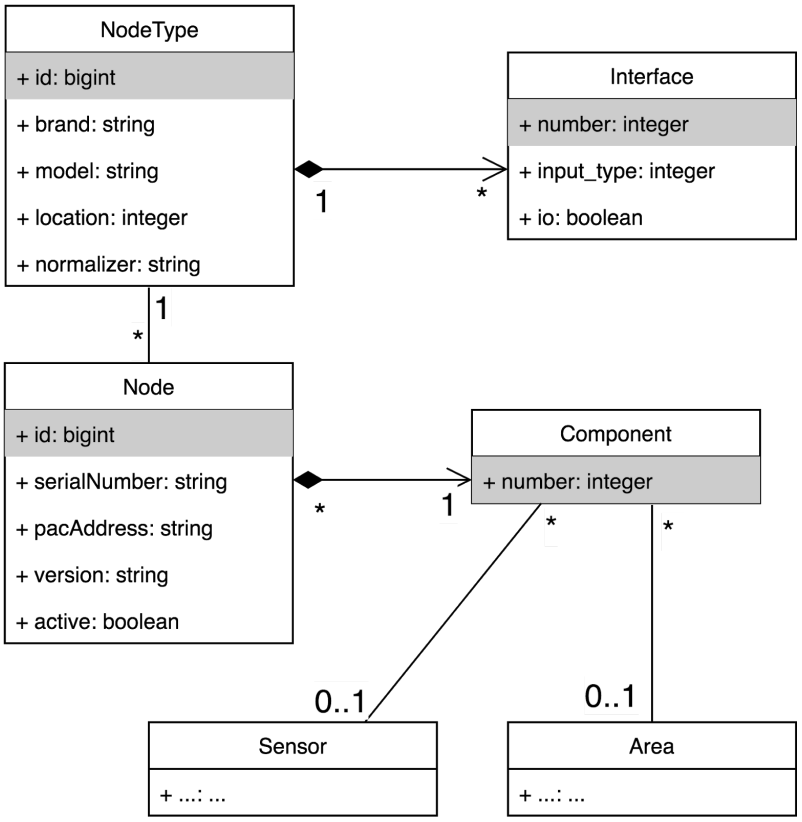


Figura 28. Especificació relacions *Component*

10.1.4 Notificacions en *Real-Time*

Una de les noves funcionalitats del backoffice és la notificació, en la mateixa pàgina web, en temps real de quan arriba una dada nova al sistema. Les notificacions s'han de produir quan:

- Arriba una request amb dades de consum.
- La request ha generat una alarma.
- La request conté dades errones.
- La request s'ha normalitzat correctament.

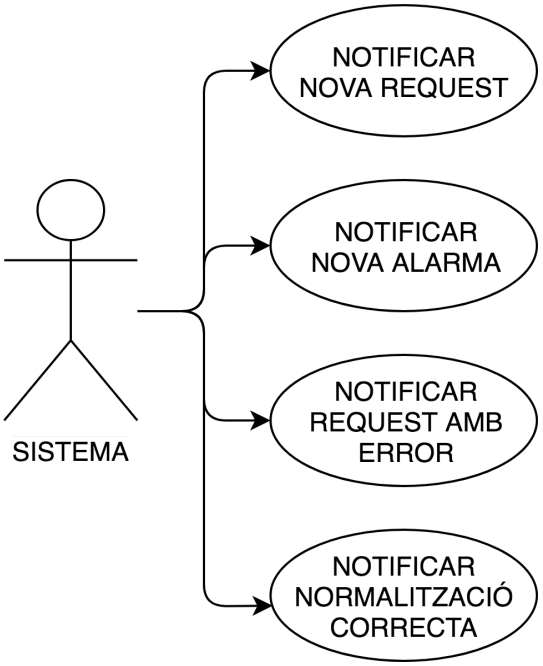


Figura 29. Casos d'ús WebSockets

Cas d'ús:	Notificar nova request	Actor:	Sistema
Precondició:	-		
Disparador:	El sistema reb una nova request		
Escenari d'èxti			
1. El sistema notifica la nova request a través dels WebSockets			
Extensions:			
-			

Cas d'ús:	Notificar nova alarma	Actor:	Sistema
Precondició:	-		
Disparador:	El procés de normalització genera una alarma		
Escenari d'èxti			
1. El sistema notifica la nova alarma a través dels WebSockets			
Extensions:			
-			

Cas d'ús:	Notificar request amb error	Actor:	Sistema
Precondició:	-		
Disparador:	El procés de normalització detecta un error en les dades		
Escenari d'èxti			
1. El sistema notifica la rebuda de dades errònies a través dels WebSockets			
Extensions:			
-			

Cas d'ús:	Notificar normalització correcte	Actor:	Sistema
Precondició:	-		
Disparador:	El procés de normalització es completa satisfactoriament		
Escenari d'èxti			
1. El sistema notifica la nova dada normalitzada a través dels WebSockets			
Extensions:			
-			

10.2 Disseny

Un cop s'han especificat les noves funcionalitats, es pot determinar el diagrama complet de classes del sistema amb les seves relacions i les restriccions que s'imposen, l'esquema de taules de la base de dades, i els diagrames de seqüència que segueixen les noves funcionalitats de l'aplicació.

10.2.1 Diagrama de classes

El diagrama de classes mostra com es relacionen cada classe i amb quines cardinalitats. A més es regeix per un conjunt de restriccions textuals que detallen el comportament del sistema per a mantenir vàlid l'estat del diagrama.

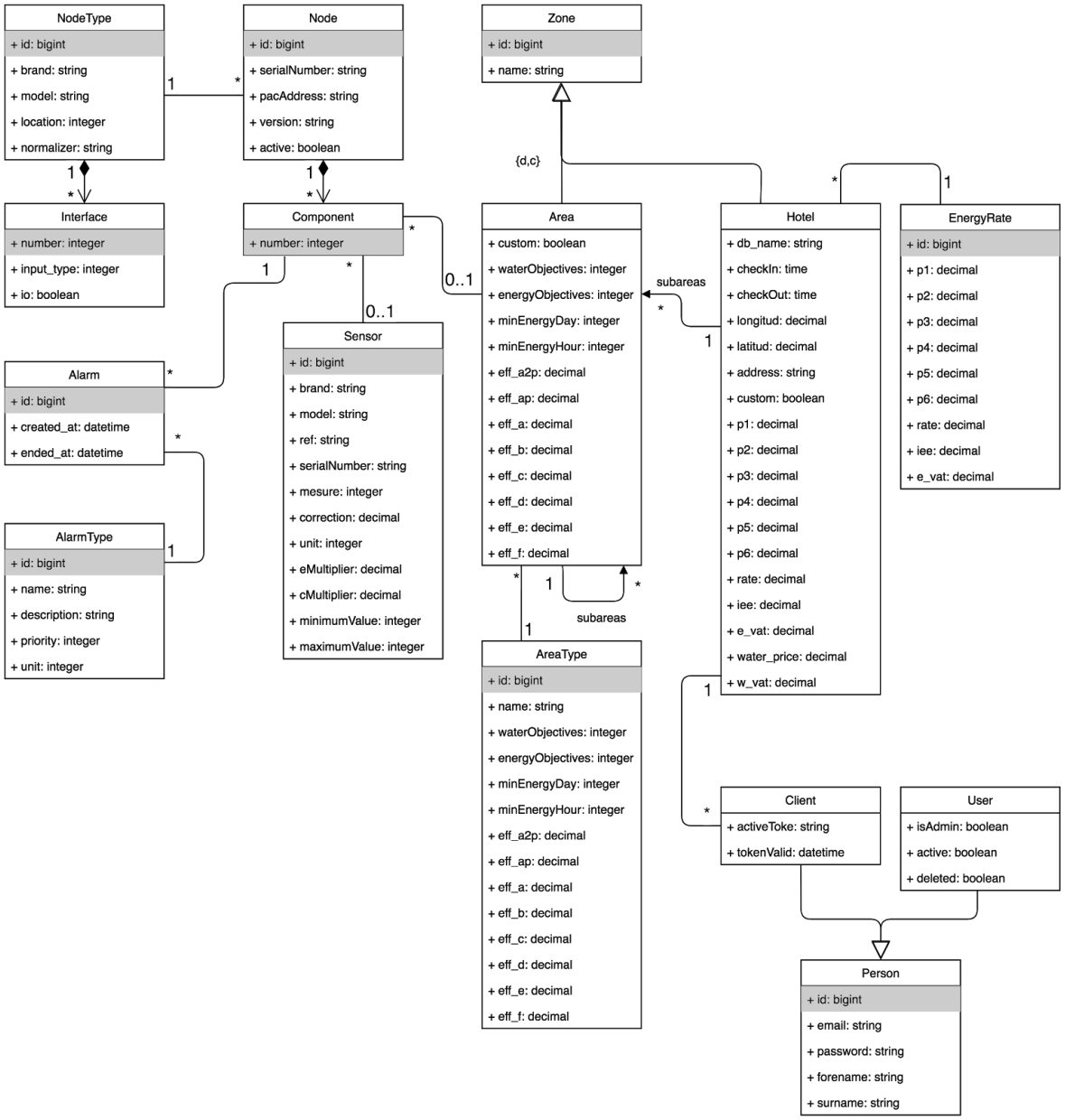


Figura 30. Diagrama de classes

Restriccions textuals

Per a resumir la llista de restriccions textuals s'ha pintat el fons d'un atribut de cada classe, a excepció de les subclasses Area, Hotel, Client i User, per a definir quines són les claus primàries de cada una d'elles.

1. La clau primària de *Interface* és *NodeType*::id - *Interface*::number
2. La clau primària de *Component* és *Node*::id - *Component*::number
3. Un *Node* ha de tenir sempre el mateix nombre de *Components* que el nombre d'*Interfaces* que té el *NodeType* associat
4. Una *Area* no pot ser subzona d'una *Area* de la qual és descendent
5. Una *Area* no pot ser subzona d'una *Area* que pertanyi a un altre *Hotel*

6. Una *Area* haurà de tenir els mateixos valors d'eficiència que els del *AreaType* associats si el camp Area::custom té valors *false*
7. Un *Hotel* haurà de tenir els mateixos valors de periode que els de l'*EnergyRate* associada si el camp *Hotel*::custom té valor *false*

10.2.2 Esquema de taules

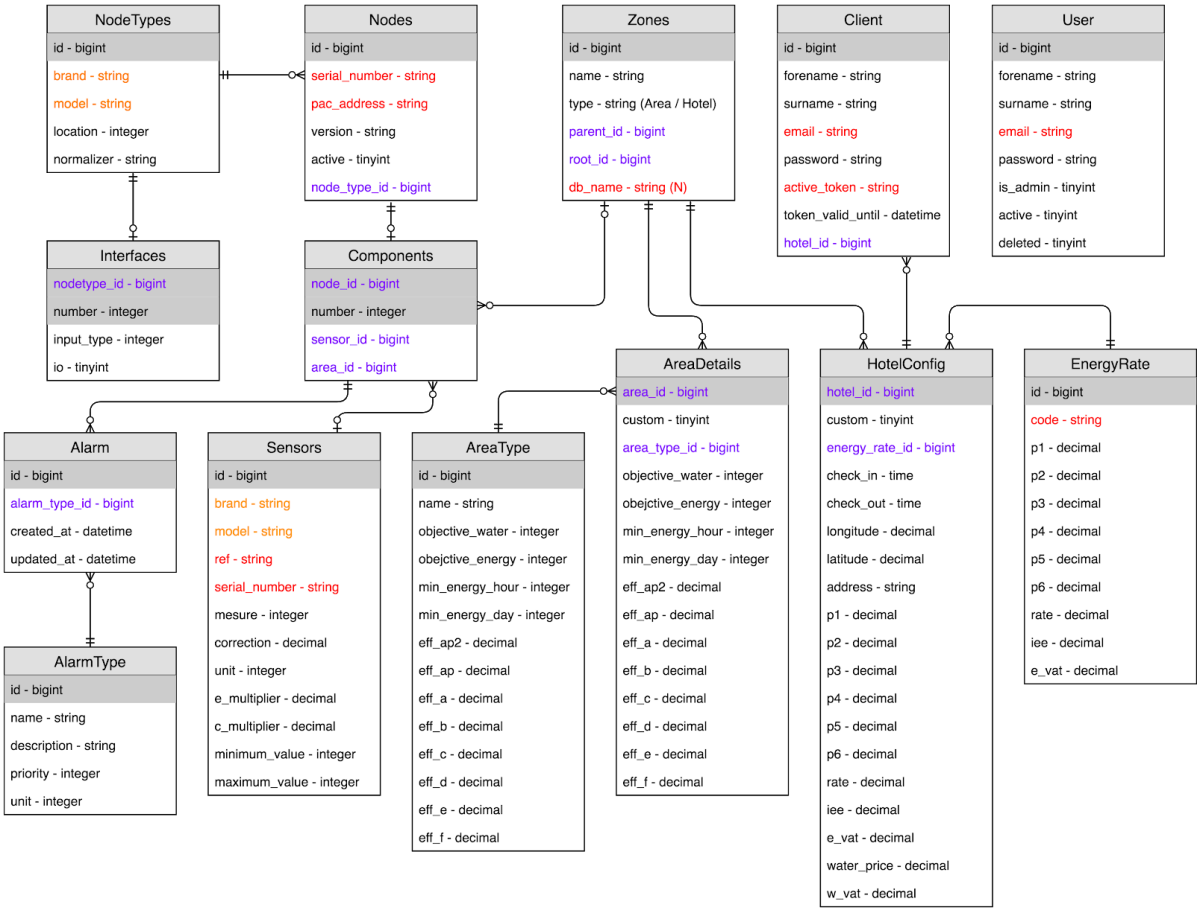


Figura 31. Esquema de taules

Per entendre en detall l'esquema de bases de dades, la següent figura mostra la llegenda del codi de colors usat per identificar les claus primàries, foranes i úniques.

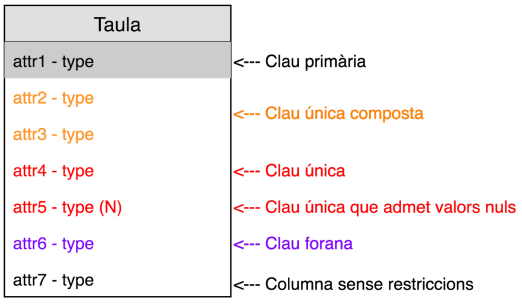


Figura 32. Llegenda esquema de taules

Si es compara l'esquema de taules amb el diagrama de classes, es poden detectar dues diferències. La primera és en la jerarquia de les classes *Zone*, *Area* i *Hotel*. I la segona amb les classes *Person*, *Client* i *User*.

En el primer cas, la traducció dels models de classes en la representació en les taules es deu a l'estratègia que s'ha fet servir per representar la jerarquia entre hotels i àrees, i entre àrees i subàrees. Aquesta és la de *Single Table Inheritance* (STI). Aquesta estratègia tracte de representar totes les possibles subclasses en una mateixa taula. En el cas de la taula *Zones*:

- L'atribut *Zone::type* serveix per identificar quin tipus de subclasse representa. Els únics valors vàlids són "*Area*" i "*Hotel*". Per tant, des de l'aplicació, per obtenir totes les instàncies d'àrea, s'han d'obtenir les files ón el valor de l'atribut *Zone::type* sigui "*Area*". I per obtenir els hotels, el valor ha de ser igual a "*Hotel*".

- Per a representar la relació de *Subareas* entre un *Hotel* i les seves *Areas*, es fa servir l'atribut *Zone::root_id*. Aquesta clau forana referència l'atribut *Zone::id*. És una referència a una fila de la mateixa taula. En el cas que es necessiti, per exemple, obtenir les àrees de l'hotel amb id 1, s'ha de fer una query a la taula buscant totes les files on l'atribut Zone::root_id sigui 1.

- En el cas de la relació de *Subareas* entre una *Area* i una d'altra, l'atribut que s'utilitza com a clau forana és Zone::parent_id. Aquest funciona de la mateixa manera que l'atribut *Zone::root_id* per a la relació *Hotel-Area*.

En el segon cas, s'ha decidit separar les classes *Client* i *User* en dues taules completament diferents. S'ha decidit realitzar-ho d'aquesta manera, ja que aquests models representen persones les quals tenen accés o bé al back-office o bé al portal, i per tant, s'ha preferit mantenir cada registre en el seu context de permisos.

10.2.3 Diagrames de seqüència

En aquest apartat es mostraran els diagrames de seqüència de cada un dels casos d'ús de les noves funcionalitats.

1. Tipus de Node

Llistar els tipus de node

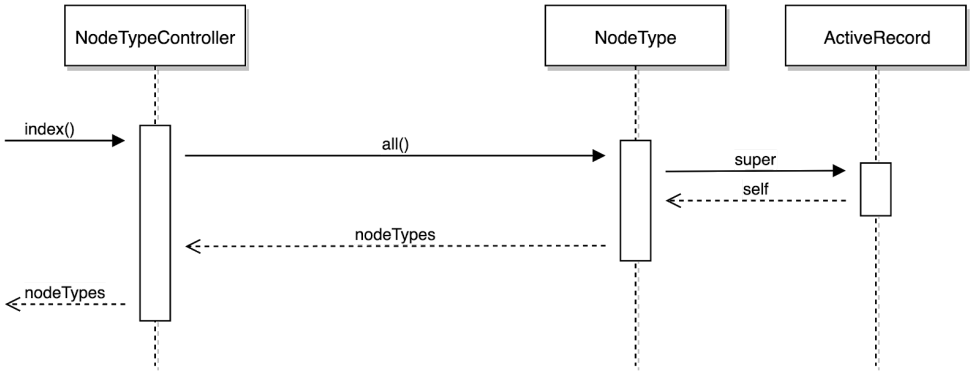


Figura 33. Diagrama de seqüència - llistar tipus de node

Llegir un tipus de node

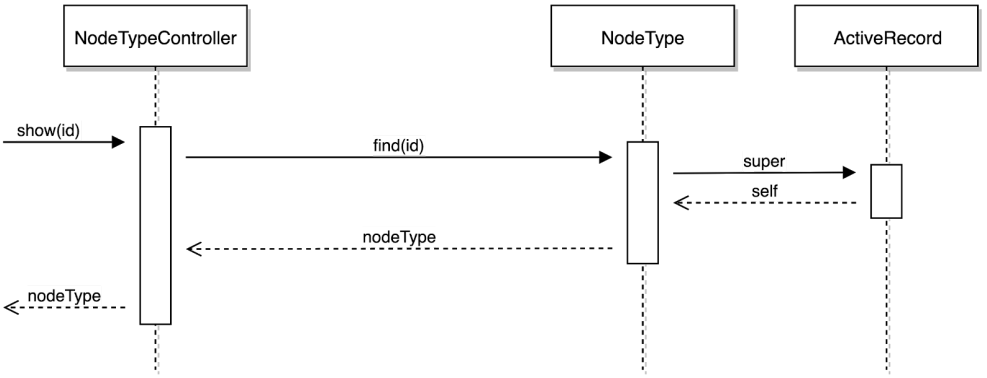


Figura 34. Diagrama de seqüència - llegir tipus de node

Crear un nou tipus de node

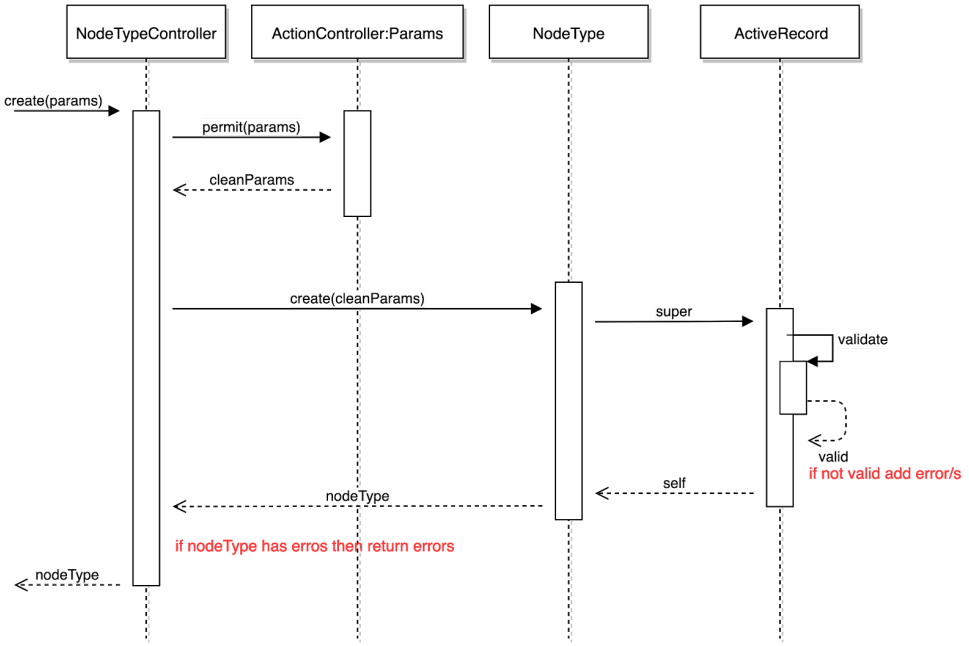


Figura 35. Diagrama de seqüència - crear tipus de node

Editar un tipus de node

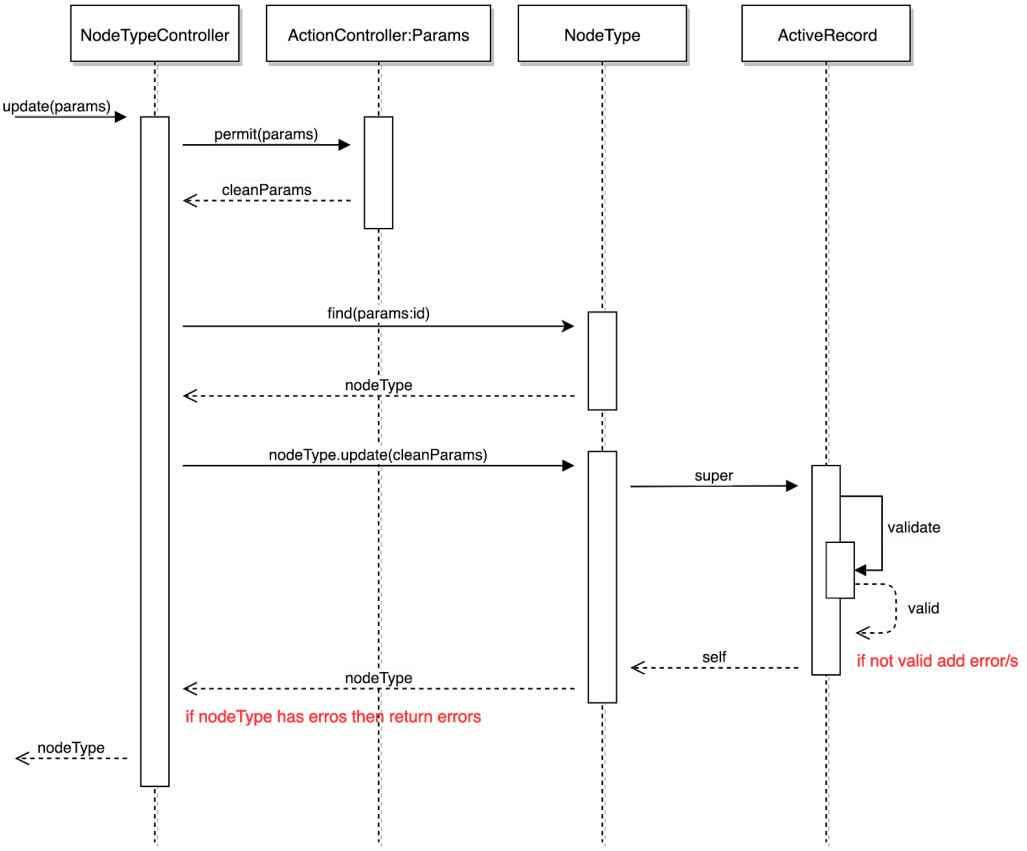


Figura 36. Diagrama de seqüència - editar tipus de node

Eliminar un tipus de node:

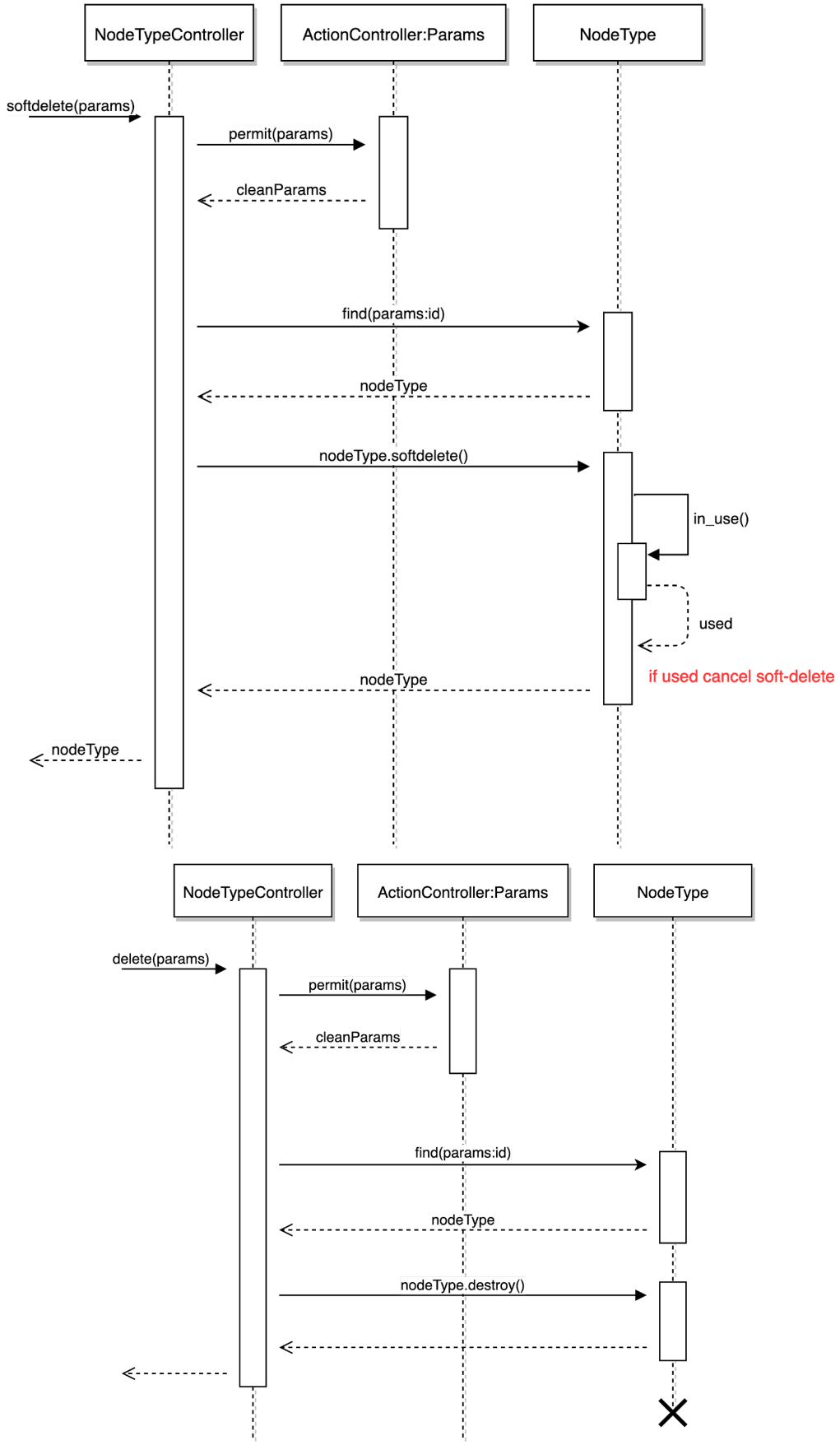


Figura 37. Diagrama de seqüència - eliminar tipus de node

Assignar un tipus a un node

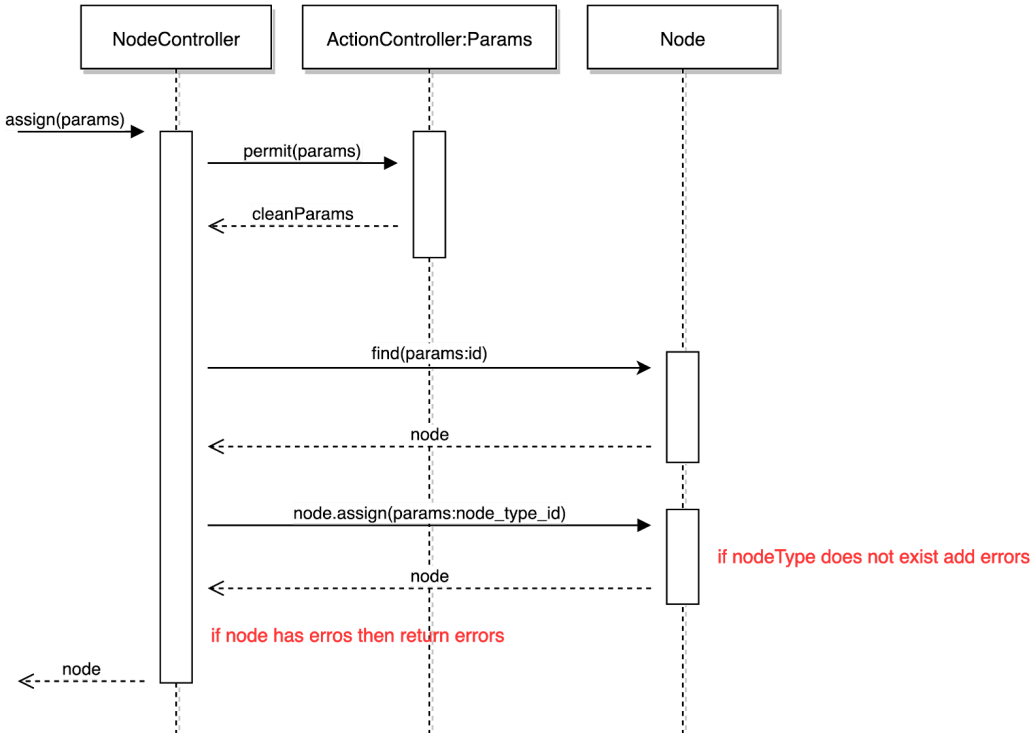


Figura 38. Diagrama de seqüència - assignar tipus de node

2. Jerarquització de les àrees

Jerarquitzar una àrea

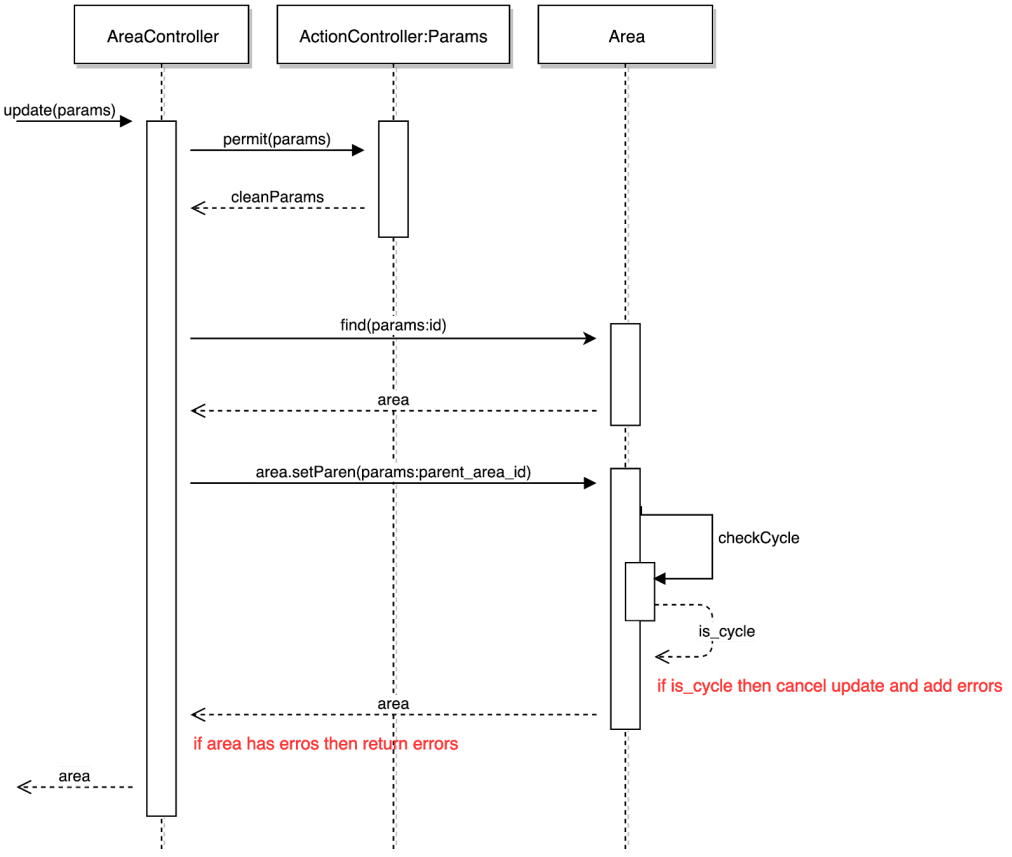


Figura 39. Diagrama de seqüència - jerarquitzar una àrea

3. Indeterminar nombre de nodes

Create Interface

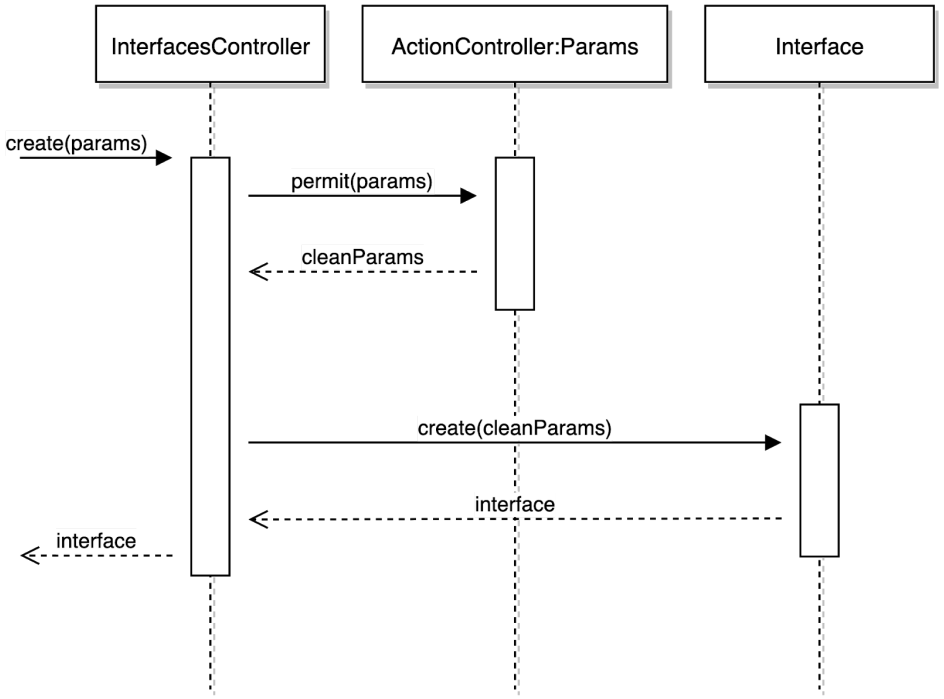


Figura 40. Diagrama de seqüència - crear interface

Update interface

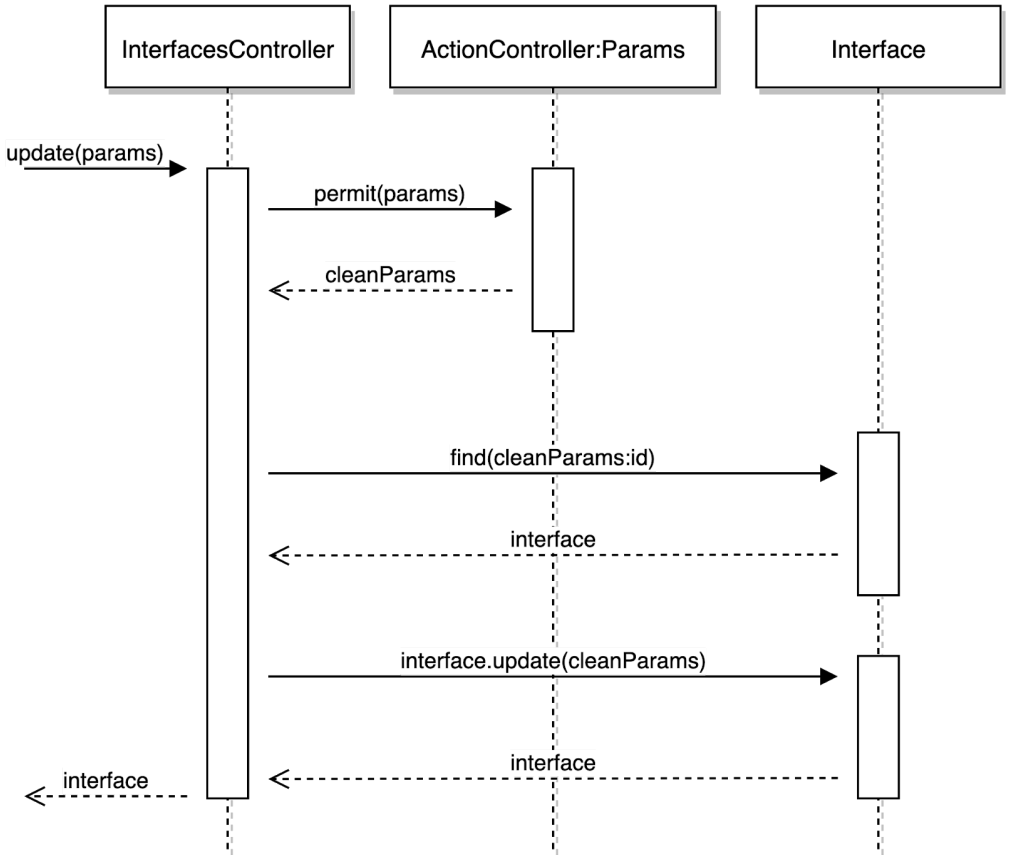


Figura 41. Diagrama de seqüència - editar interface

Delete interface

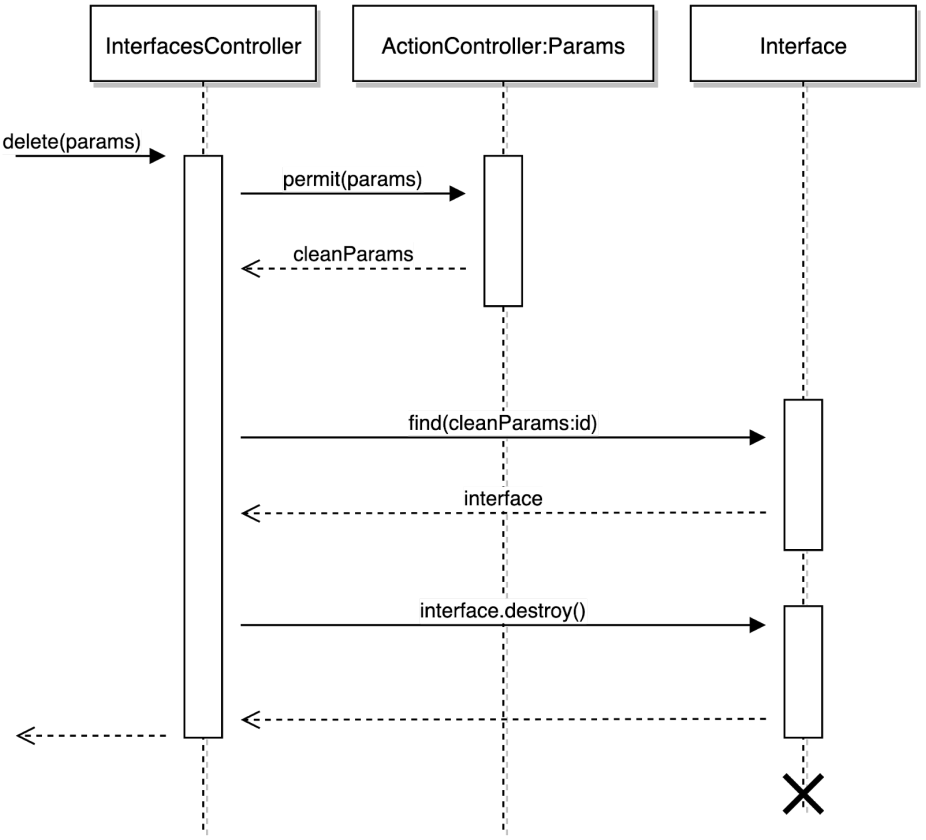


Figura 42. Diagrama de seqüència - eliminar interface

Connect sensor to component

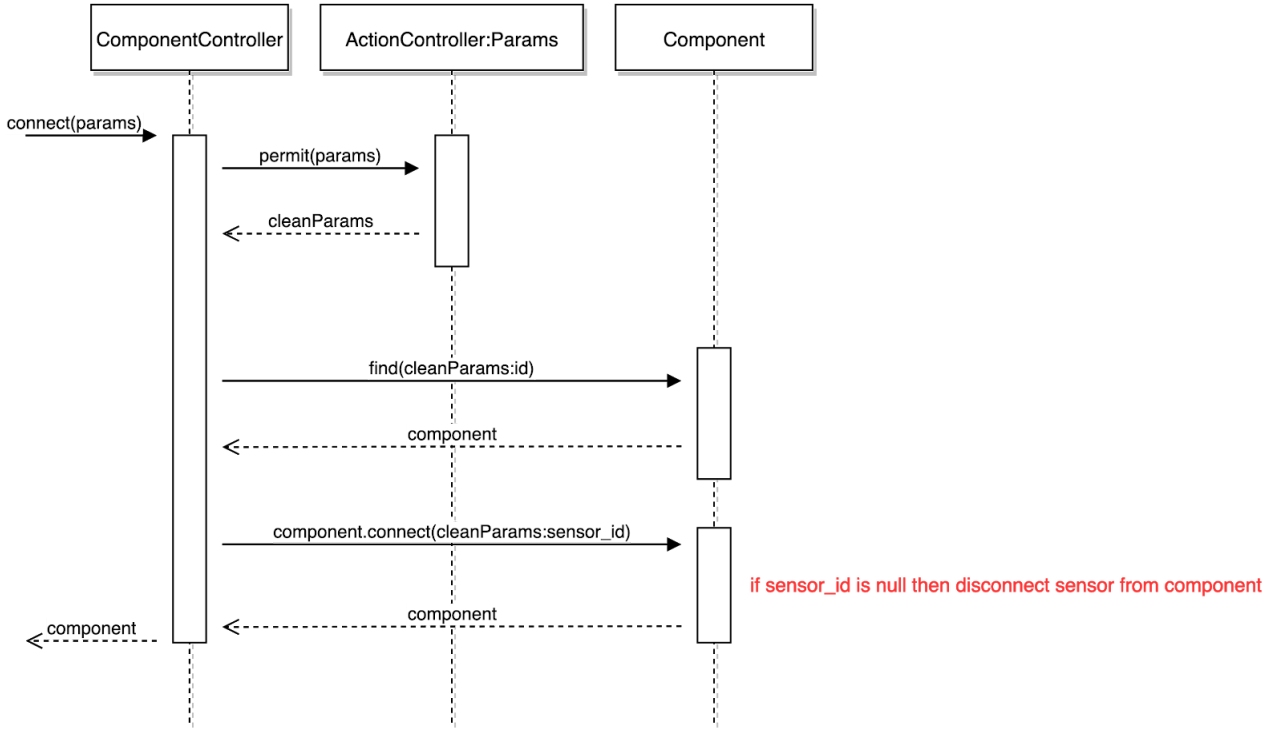


Figura 43. Diagrama de seqüència - conectar sensor a un component

Assign area to component

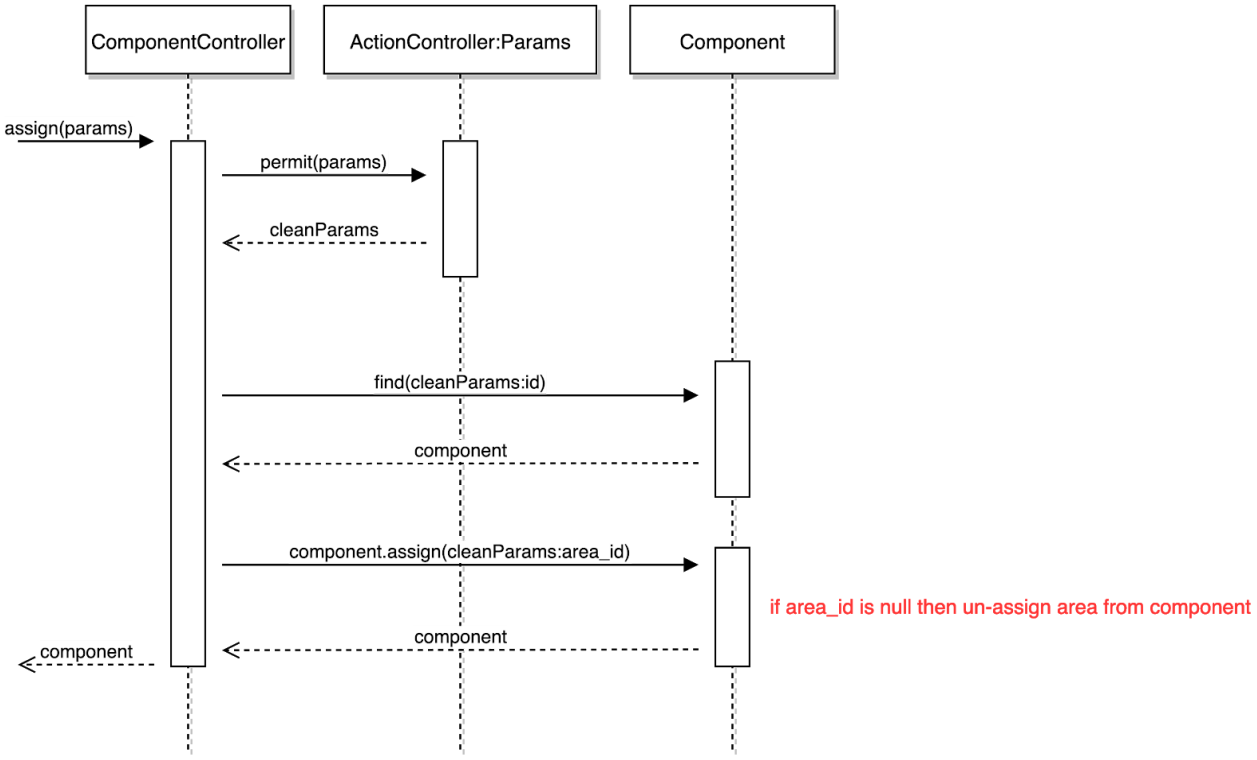


Figura 44. Diagrama de seqüència - assignar àrea a un component

4. Notificacions en temps real

Notificar nova request

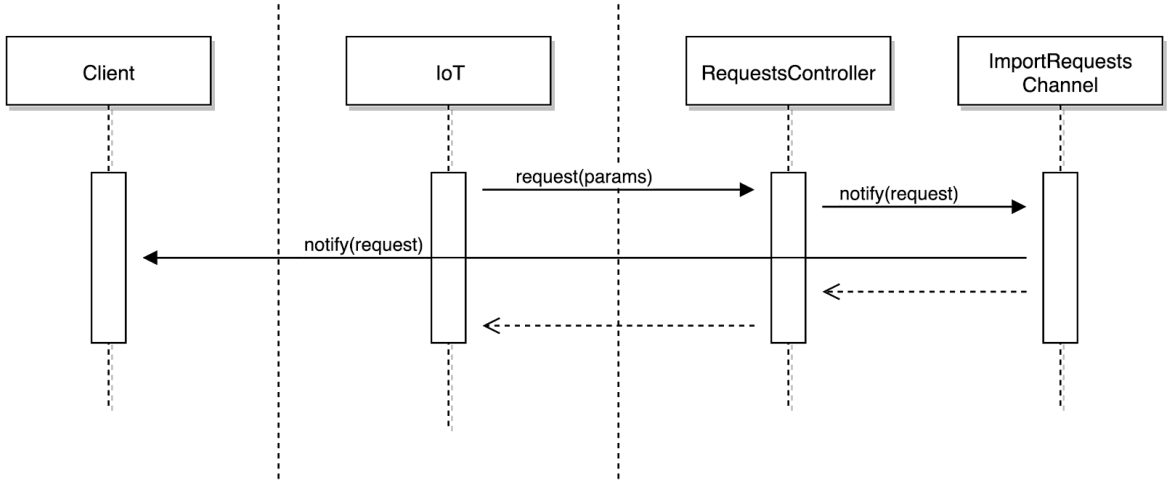


Figura 45. Diagrama de seqüència - notificar nova request

Notificar alarma

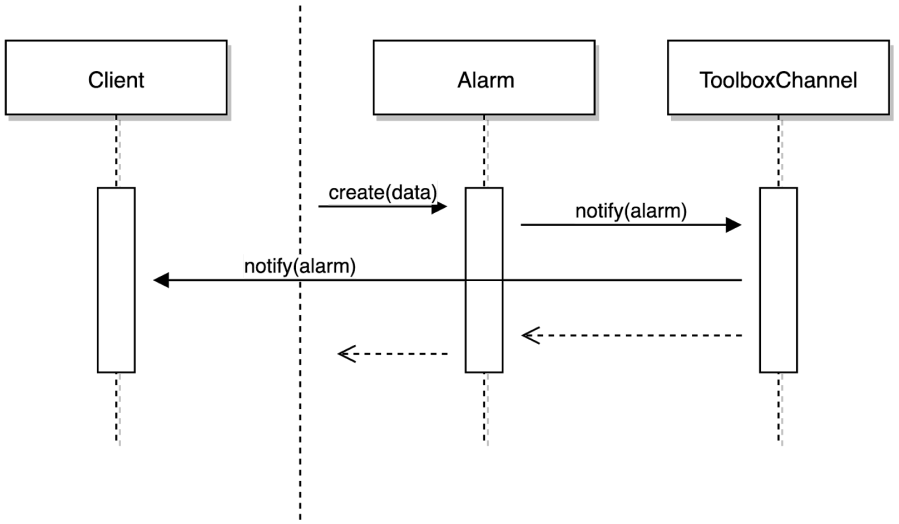


Figura 46. Diagrama de seqüència - notificar nova alarma

Notificar request normalitzada correctament / amb error

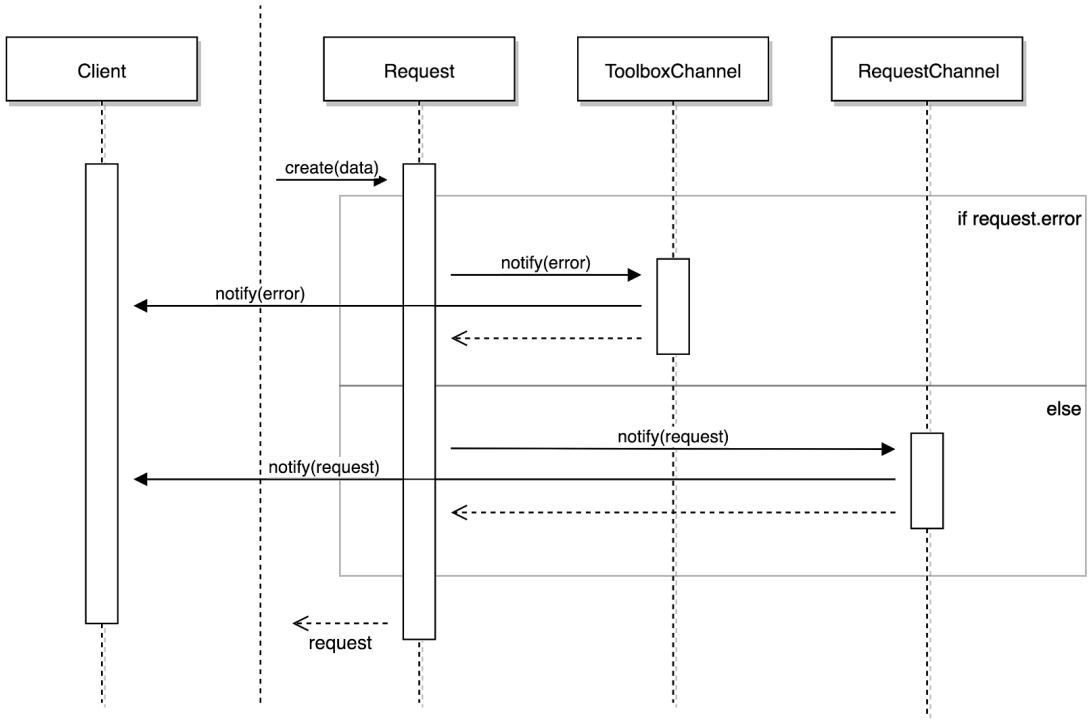


Figura 47. Diagrama de seqüència - notificar normalització

5. Interacció Front-end i Back-end

En tots els diagrames, l'execució es realitza com a resposta a l'acció de l'usuari. La interacció entre el front-end i el back-end es dur a terme a través de la següent seqüència de passos:

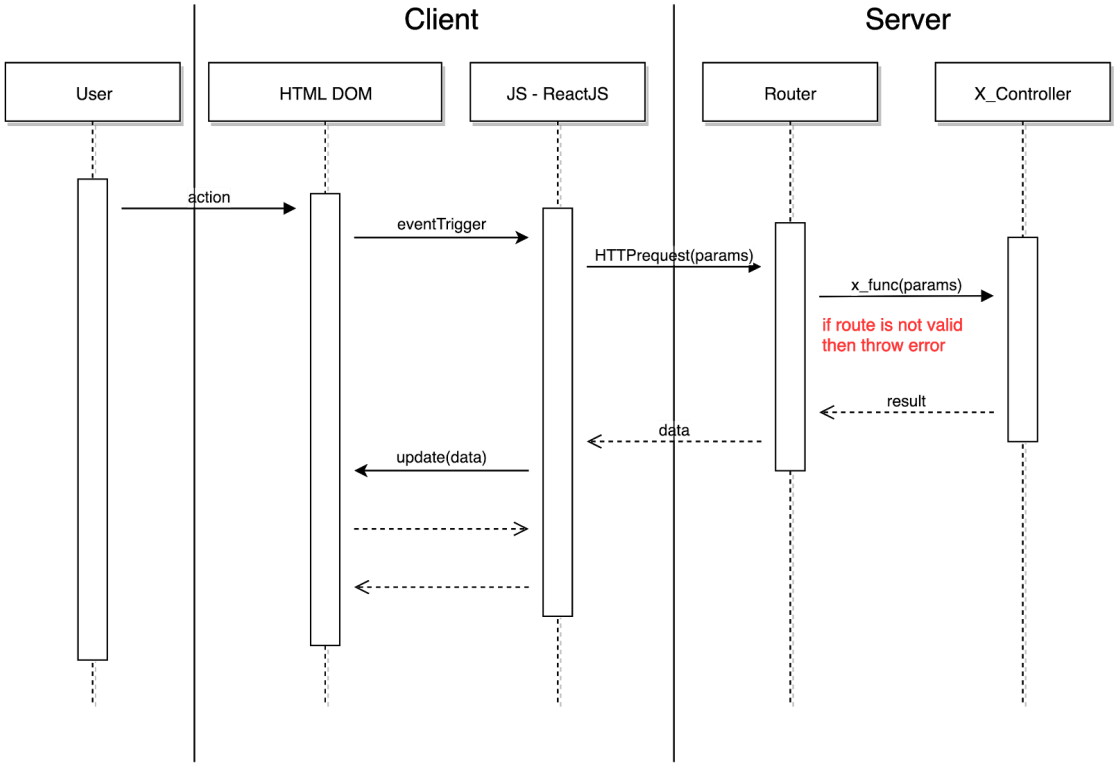


Figura 48. Diagrama de seqüència - interacció front-end i back-end

El router ha de resoldre la request HTTP i executar la funció assignada del controlador corresponent. En cas que la ruta no existeixi, retornarà un error al client.

10.2.4 Interfície

Les noves funcionalitats afegeixen nous panells de control o en modifiquen d'existents afegint-hi o canviant-ne camps en els formularis. Per a mantenir un aspecte continu entre les noves vistes i les resultants del redisseny de la interfície, les noves segueixen el mateix patró de *layouts* i *templates*. En aquest sentit, les noves funcionalitats no incorporen canvis en el propi disseny de les interfícies.

10.3 Implementació

Les noves funcionalitats han incorporat canvis tant en l'arquitectura del sistema com en el diagrama de components resultant del redisseny.

Les implementacions de les funcionalitats Tipus de Node, Jerarquització de les àrees i Indeterminar el nombre de sensors per node, han seguit el mateix procés d'implementació dut a terme durant l'etapa del redisseny per a la Web App i usant les tecnologies especificades. Aquestes funcionalitats no han incorporat cap nou mòdul en l'arquitectura, però sí que han creat nous components en el back-end i en el front-end.

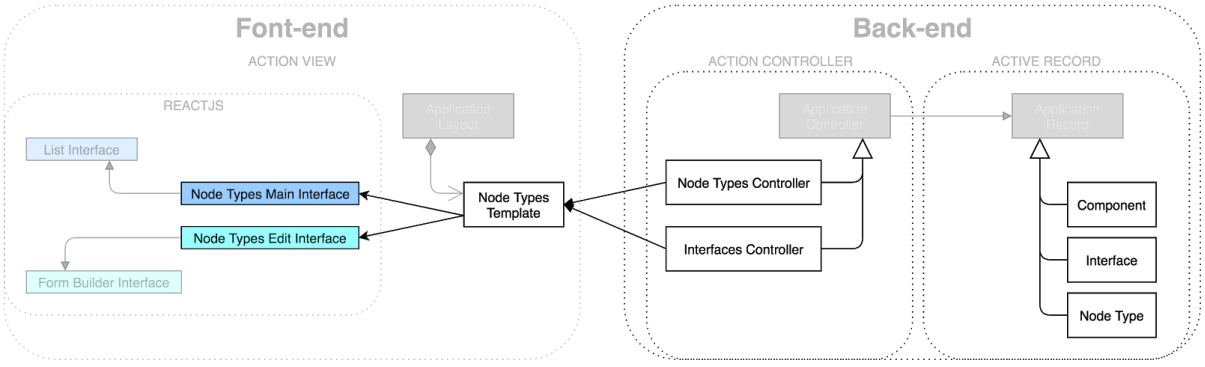


Figura 49. Diagrama de components noves funcionalitats

10.3.2 Implementació de les Notificacions en temps real

La implementació de les notificacions en temps real ha implicat, a més d'afegir nous components, canvis en l'arquitectura i s'ha usat una nova tecnologia. Per a dur-la a terme s'han utilitzat connexions entre el client i el servidor de tipus WebSockets.

Un WebSocket (WS) és una connexió TCP bi-direccional entre un client i un servidor. A diferència de les connexions HTTP en les que el client és l'únic que pot inicialitzar una comunicació, en un WS, el servidor pot enviar dades noves als clients connectats [34] [35]. Les comunicacions funcionen sobre un sistema de subscripcions o cada subscripció té un canal de transmissió. El servidor només podrà comunicar-se amb els clients que s'hagin subscrit al canal per on es vol transmetre informació.

El framework utilitzat, RoR, ja incorpora una API interna per a facilitar l'ús dels WS en una aplicació. Per a l'ús d'aquesta API, RoR incorpora dues carpetes en el seu arbre de fitxers: */app/channels* i */app/assets/javascripts/channels*. Cada una conté els fitxers amb la lògica de cada punt de la connexió servidor-client respectivament.

/app	Conté tot el contingut de l'aplicació en si
/assets	
/javascripts	Conjunt de fitxers Javascript que s'executaran al client
/channels	*End-point* del client en les comunicacions via WS
/...	
/channels	*End-point* del servidor en les comunicacions via WS
/...	
/...	

Taula 30. Estructura del codi WS

Tot i això, per a poder funcionar correctament, una connexió d'aquest tipus necessita, per part del servidor, un sistema de control de flux de dades. En aquest projecte s'ha utilitzat Redis. Redis és una eina open-source per a la iimplementació d'estructures de dades key-value amb una durabilitat fixa. En el cas del projecte funciona com a caché i com gestor de cues de peticions [36]. És a dir, quan arriba una nova petició, aquesta s'afegeix a una cua virtual fins que pugui ser processada. En l'arquitectura del sistema, Redis es comunica amb el backoffice bidireccionalment.

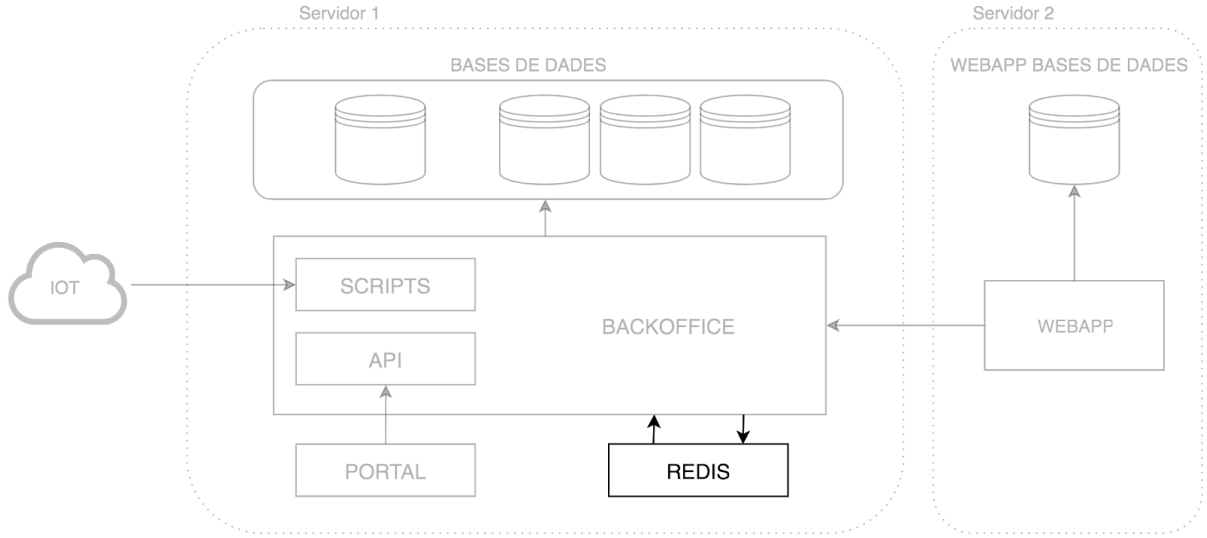


Figura 50. Diagrama definitiu de l'arquitectura

El procés d'implementació dels websockets s'ha realitzat amb tres fases. La primera és garantir que el punt de connexió entre el servidor i el client funciona correctament. En el cas de l'entorn de desenvolupament, la màquina virtual simula el comportament de Redis o eines similars. En el cas de l'entorn de test, primer s'ha hagut d'instal·lar el servei redis en el servidor i configurar-lo amb les opcions desitjades. En el cas d'aquest projecte, s'han mantingut la configuració per defecte.

La segona és habilitar els *websockets* a l'aplicació web. RoR té un mòdul intern, *ApplicationCable*, que conté dues classes: *Connection* i *Channel*. La classe *Connection* s'encarrega de la persistència de les connexions i en garanteix que aquestes es realitzin amb els permisos adients. La classe *Channel* controla les subscripcions que el client realitza i és el port d'entrada i sortida de missatges entre el client i el servidor. Els *websockets* no utilitza HTTP, sinó que té el seu propi protocol: WS (o WSS respecte HTTPS). Per inicialitzar la connexió el client s'ha de subscriure a un dels canals disponibles amb una petició HTTP a la ruta "/cable".

La tercera és habilitar el client. Quan es crea un *Channel* en RoR seguint les pautes de la documentació de RoR, també es crea un objecte javascript que gestionarà les subscripcions que realitzi el client a aquell *channel*. L'objecte tindrà dues funcions que s'utilitzen com a port de sortida i de rebuda de missatges per part del client. Com que en la capa de vistes de l'aplicació s'utilitza un *layout* amb diferents *partials* i un *templete* amb més d'un component react, es donen casos on es realitzen múltiples connexions a un mateix canal. Per evitar connexions paral·leles, s'han modificat els objectes javascript per ser accessibles només des d'una interfície. Quan s'inicialitza la pàgina, es crea una connexió i cada component que necessita accedir-hi, se subscriu a la interfície, sense crear cap connexió nova.

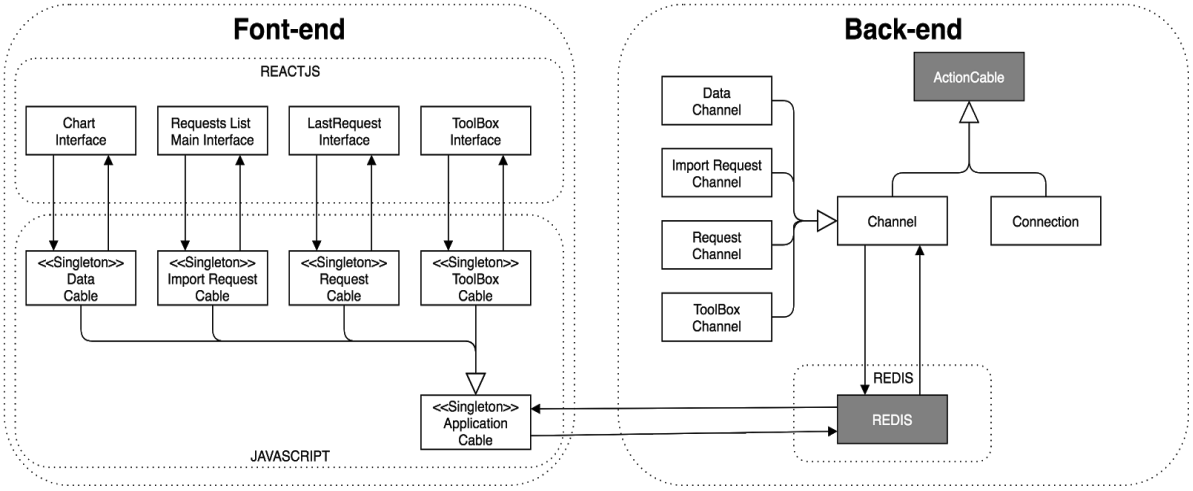


Figura 51. Diagrama de components dels WebSockets

11 Pla de proves

En aquest projecte s'hi ha volgut atribuir importància al pla de proves, ja que un dels principals motius que va impulsar la realització d'aquest és millorar el sistema actual en robustesa i en tolerància a errors, entre d'altres. Aquest capítol es divideix entre els tests de requisits funcionals i no funcionals, on els segons tenen un pes més alt.

11.1 Proves pels requisits funcionals

Per a comprovar que es compleixen tots els requisits funcionals, s'han realitzat diferents tests segons els escenaris possibles de cada cas d'ús. En total s'han realitzat 106 tests, dels quals 37 pertanyen a les noves funcionalitats. Tot i això, en aquest apartat només es mostraran els tests de dos casos de casos d'ús com a exemples del mètode usat en la definició dels tests.

1. Cas d'ús jerarquització d'una àrea.

   Aquest cas d'ús presenta dos requisits funcionals que determinen el comportament de l'operació segons l'estat de les dades d'abans de realitzar-la. El primer diu que "*Dues àrees només poden estar jerarquitzades entre si, independentment del nombre de nivells intermedis, si les dues àrees pertanyen al mateix hotel*", i el segon: "*La jerarquització de les àrees no pot crear cicles: Una àrea A no pot categoritzar-se com a subàrea d'una àrea B, si aquesta és, independentment del nombre de nivells intermedis, subàrea d'A*".

   Donades les dues casuístiques, del primer requisit en surten dos tests:

   Test 1: Es volen jerarquitzar dues àrees del mateix hotel. En aquest cas l'operació pot continuar executant-se, ja que compleix el requisit.

   Test 2: Es volen jerarquitzar dues àrees d'hotels diferents. L'operació s'ha d'avortar, ja que incompleix el requisit. Segons la descripció del cas d'ús, s'ha de mostrar un missatge d'error informant l'usuari què és el que ha fallat.

   I del segon requisit:

   Test 3: Donades les àrees A i B. A i B no tenen cap relació de jerarquia. En aquest cas l'operació pot continuar executant-se, ja que compleix el requisit.

   Test 4: Donades les àrees A i B. Es vol fer que B sigui subàrea d'A on A ja és subàrea de B. Si A és subàrea de B, l'operació s'ha d'avortar, ja que, en cas contrari, es crearia

un cicle en la jerarquia d'àrees. S'ha de mostrar un missatge d'error a l'usuari informant de l'error.

Test 5: Donades les àrees A i B. Es vol fer que B sigui subàrea d'A on B ja és subàrea d'A. Aquest cas no s'incompleix el requisit, ja que si B és subàrea d'A, amb aquesta operació només es modificaran el nombre de nivells intermedis.

2. Cas d'ús connectar un sensor a un component.

   El cas d'ús té un requisit específic que limita els components a on es pot connectar un sensor. Més concretament, només es pot connectar-ne als components amb un port d'entrada i de tipus digital. D'aquest punt en sorgeixen tres tests:

   Test 1: Es vol connectar un sensor a un component amb un port d'entrada digital. Aquest és l'únic cas que permet l'execució completa del cas d'ús.

   Test 2: Es vol connectar un sensor a un component amb un port d'entrada no digital. En aquest cas s'ha de mostrar un missatge d'error i avortar l'operació.

   Test 1: Es vol connectar un sensor a un component amb un port de sortida digital. S'ha d'avortar l'operació i mostrar un missatge d'error.

11.2 Proves pels requisits no funcionals

L'avaluació dels requisits no funcionals s'ha realitzat mitjançant diferents mètodes. Cada un pensat per arribar a una conclusió més encertada depenent del tipus de requisit.

Requisit de la completesa de les funcionalitats

Tots els objectius marcats i tasques descrites han de poder dur-se a terme amb alguna de les funcionalitats del sistema.

Per a cada model existent en el sistema, hi ha una operació que permet realitzar les operacions bàsiques (CRUD) més les operacions d'associació amb els models que hi guarda relació, incloent-hi les relacions recursives del model Area. Amb aquest joc d'operacions es pot arribar a obtenir qualsevol estat correcte i possible de les dades.

Requisit de l'apropiativitat de les funcionalitats

Les funcionalitats han de ser plantejades de manera que l'usuari pugui seguir una successió de passos clars, excloent-hi passos innecessaris.

Tot i que aquest requisit manté un fort lligam amb la presentació de la interfície, es pot justificar que, independentment de l'estètica, l'usuari ha de ser capaç d'entendre com

realitzar una operació. Per aconseguir-ho, a tots els panells s'hi troben botons que s'identifiquen tant per l'etiqueta, que descriu, amb un parell de paraules com a màxim, l'acció que real que durà a terme. A més, cada operació té assignat un codi de colors que es manté en tot el projecte:

- El color verd per a indicar operacions de caràcter positiu: Generen nou contingut o en modifiquen d'actual sense presentar cap risc.

- El color groc per a indicar operacions amb cert risc i que requereixen la confirmació necessària per part de l'usuari.

- El color vermell per a indicar operacions amb risc alt i que també requereixen confirmació de l'usuari.

- El color blau per a indicar que l'operació només mostra informació útil a l'usuari i no té cap efecte en l'estat de les dades.

Requisit de temps de resposta

Una acció ha de respondre amb un temps apropiat a la complexitat de la tasca que realitza.

Per a valorar el temps de resposta de l'aplicació, s'han agafat operacions que s'agrupen segons del tipus de dades que transmeten en tres grups. Les primeres fan una request HTTP a través d'una crida interna del front-end on s'envia i es reben dades en format JSON. Les segones són requests de documents: carregen un document HTML amb totes les dependències JS i CSS d'aquest. Les últimes són crides HTTP a través de l'api realitzades des d'un punt de connexió extern.

En la taula es mostren els resultats de sis operacions, dos de cada tipus, del total de les proves realitzades. Tots els valors estan representats en milisegons (ms).

Tipus	Operació	T1	T2	T3	T4	T5	Tavg
data/json	Jerarquitzar area	57	41	51	42	45	47.2
data/json	Assignar àrea a component	48	42	53	47	44	46.8
document	Carregar *HomePage*	1930	1960	1775	1820	1850	1867
document	Carregar *RequestsPage*	1600	1680	1550	1570	1480	1576
api	Login	278	164	159	183	163	189.4
api	Obtenir dades hotel	416	637	415	485	498	490.2

Taula 31. Temps de resposta

L'objectiu d'èxit del requisit s'havia marcat als 2 segons de temps de resposta com a màxim. Observant els resultats, es pot comprovar que cap d'elles supera els dos segons, tot i que alguna càrrega de document s'hi apropa.

Requisit d'aprenentatge autònom

El sistema ha de ser intuïtiu perquè un usuari pugui utilitzar-lo des d'un inici sense formació i sigui fàcil recordar com realitzar les mateixes accions en el futur.

Per a comprovar els requisits s'ha realitzat una prova a 20 persones amb perfils diversos sense cap coneixement previ de l'aplicació i la distribució de les pàgines. Aquesta prova consisteix a realitzar 10 operacions que busquen conèixer si l'usuari és capaç d'aprendre a realitzar-la per primera vegada i de recordar-se'n els cops posteriors.

Les operacions són:

1. Crear un hotel nou
2. Crear un nou tipus de node amb mínim dues interfícies
3. Crear un nou node i assignar-hi el tipus creat al pas anterior
4. Connectar sensors als components del node
5. Crear dues àrees a l'hotel
6. Assignar els components a les àrees
7. Crear un nou node d'un tipus nou
8. Crear una àrea nova
9. Jerarquitzar les àrees sota l'última creada
10. Crear un client perquè es pugui connectar al portal.

Un cop realitzades les operacions, han hagut de respondre 4 preguntes donant una valoració de l'1 al 5.

Pregunta	Valor de l'1	Valor del 5	Resultat
Sense tenir cap coneixement del sistema, amb quin grau de dificultat avaluaries les tasques requerides?	Trivials	Difícils	2
Un cop realitzades les tasques. Com consideres la interfície?	Intuitiva	Laberintica	2
En cas de realitzar novament alguna de les tasques. Com creus que reaccionaries?	Ho puc fer amb els ulls tancats	Necessitaria ajuda d'algú	3
Estaries d'acord en la necessitat de l'existència d'una guia o manual d'usuari?	Molt en desacord	Molt d'acord	3

Taula 32. Enquesta requisit N.F. d'aprenentatge autònom

Requisit de protecció a l'usuari

El sistema ha de guiar a l'usuari per tal de reduir i/o eliminar errors innecessaris i/o evitables.

El sistema emmagatzema les contrasenyes dels usuaris i dels clients. Aquesta informació s'ha de mantenir en la màxima confidencialitat i seguretat. Per aconseguir-ho, RoR, amb l'ajuda de Devise, encripta les contrasenyes. El procés d'encriptació no permet desfer el resultat, pel que és impossible conèixer el valor inicial previ a l'encriptació.

Requisit d'estètica de la interfície

La interfície ha de ser neta i ha d'estar ben estructurada.

Per a comprendre l'estètica de la interfície, s'ha fet un qüestionari a les mateixes 20 persones que van respondre per avaluar la completesa del requisit d'aprenentatge autònom.

Pregunta	Valor de l'1	Valor del 5	Resultat
Distribució dels elements en la pantalla	Dolenta	Bona	4
Densitat d'elements en la pantalla	Densa	Buida	3
Mida dels elements	Massa petits	Massa grans	3
Colors emprats	Discordants	Homogenis	3
Responsive (que s'adapta correctament a dispositius mòbils)	En desacord	D'acord	4

Taula 33. Enquesta requisit N.F. d'estètica de la interfície

Amb aquests resultats es pot concloure que l'aplicació presenta una interfície clara on cada element té un espai correcte i distingible. A més, la concordança de colors afavoreix la lectura i no distreu a l'usuari.

Requisit de tolerància d'error

El sistema ha de programat per a resistir qualsevol error que pugui produir-se.

El framework de RoR, té eines que s'encarreguen d'evitar que el sistema falli. Els possibles punts febles, en el back-end de l'aplicació, on es poden concentrar els errors s'han resumit en tres seccions:

- Connexions externes del sistema: IoT, WS i API. Aquest àmbit s'encara en la seguretat del sistema. Per a realitzar de forma segura les connexions amb elements externs o amb protocols diferents de l'HTTP es basa la seguretat en els següents elements. En primer lloc el servidor de proves consta d'un certificat SSL que regula el trànsit en

connexions HTTPS i WSS, fent que aquestes estiguin encriptades. A més el propi framework de RoR, amb l'ajuda de Device, realitza un control de permisos i sessions. Prèviament a qualsevol operació, a excepció de les de logIn i la rebuda de dades, es realitza una comprovació per saber si el client té una sessió vàlida oberta. En cas contrari es bloqueja l'operació.

- Controladors. Els controladors tenen dues eines de detecció i processament de les peticions que permet fer un control d'errors més exhaustiu: el router de funcions i en el mòdul de gestió de paràmetres. El router té la funció de resoldre els URLs de les peticions i redirigir les que tenen un url no permès o no conegut. El mòdul de paràmetres filtra els paràmetres requerits i/o permesos, i els pot transformar en el format corresponent.

- Estat de les dades. La regulació de l'estat de les dades i la seva modificació es relega al mòdul de validacions d'ActiveRecord i al mòdul de connexió de mySQL per a convertir de forma segura les peticions en queries SQL. Aquests dos mòduls interns s'encarreguen de fer les validacions pertinents prèvies en les operacions de lectura i escriptura de dades per evitar-ne errors en la persistència.

En el cas del front-end, el que ha de gestionar els errors és ReactJs. El framework és responsable de donar una resposta a les interaccions de l'usuari en el client. En tot moment, cada component ha de mantenir un estat de dades vàlid i amb concordança amb la resta. Per fer el control d'errors, les dades s'han estandarditzat perquè sempre tinguin un valor per defecte i evitar comportaments no desitjats, i en última instància errors.

Requisit de confidencialitat

Cada usuari ha de poder guardar dades privades.

En aquest aspecte, la confidencialitat de les dades no són dades personals dels mateixos usuaris sinó que són les dades de consums de cada hotel. Per raons òbvies, un hoteler només ha de poder veure i analitzar les dades de consum d'aigua i electricitat de les greenrooms del seu hotel, i ha de tenir la certesa absoluta que ningú extern a l'hotel, a excepció del propi equip de GreenCustomers, té accés a les dades. Per a complir el requisit, s'ha desenvolupat l'API, únic punt d'accés de lectura de les dades, perquè es comprovi si les dades requerides pertanyen a l'hotel on el client està associat. Qualsevol petició fora d'aquests paràmetres es bloquejarà amb un missatge d'error.

Requisit de responsabilitat

Cada entitat ha de contenir un atribut únic a la classe per tal de poder-la identificar en qualsevol moment.

La justificació del requisit es pot realitzar simplement observant tant el diagrama de classes com el diagrama de taules de la base de dades. En els dos es pot observar com cada classe i taula té un atribut que permet identificar cada instància de forma única.

Requisit de modularitat

Cada component del sistema ha d'actuar sobre l'entitat per la qual s'ha dissenyat.

De la mateixa manera que el requisit de responsabilitat es pot justificar amb els diagrames de classes i taules. El requisit de modularitat es justifica amb el diagrama de components. Per un costat tenim que cada classe té associat un component encarregat de representar el model i un controlador encarregat de dur a terme les accions sobre l'entitat. Per l'altre costat, cada model només opera sobre l'entitat que representa i cada controlador actua sobre l'entitat el qual s'està aplicant l'acció.

Requisit de canviabilitat

El sistema ha de poder evolucionar.

El projecte s'ha desenvolupat utilitzant una metodologia àgil, la qual ha permès crear una aplicació per sprints. Al final dels quals s'obtenia un sistema completament funcional i operatiu. Per aquest motiu, es pot considerar cada sprint com un canvi a l'aplicació per afegir noves funcionalitats. El mateix mètode s'hauria pogut usar per canviar el comportament de funcionalitats existents. En conseqüència, en la possible planificació de nous sprints per fer evolucionar el sistema, aquest respondria favorablement als canvis introduïts.

12 Guia d'instal·lació del Software

En aquest apartat s'explicarà pas per pas el procés d'instal·lació de l'aplicació, usant un terminal, en un entorn similar a l'usat en l'entorn de proves del projecte (una màquina UNIX amb Debian9).

12.1 Preparació del sistema [37]

El primer pas és comprovar que el servidor està actualitzat per a les versions de les tecnologies utilitzades. Per a fer-ho s'han d'executar els següents comandes:

```
> apt-get update
> apt-get upgrade
```

Per a continuar amb la instal·lació, és necessari els programes *curl* i *gpg*, i és recomanable tenir també el paquet de software *build-essentials*. Aquests paquets es poden instal·lar amb la comanda:

```
> apt-get install curl gnupg build-essentials
```

12.1.1 RVM

Per a la gestió de versions de ruby, s'utilitza RVM (Ruby Version Manager). Aquest programa facilita el canvi de versió de ruby. Per instal·lar-ho, s'ha d'executar la següent comanda:

```
> sudo gpg --keyserver hkp://keys.gnupg.net --recv-keys
409B6B1796C275462A1703113804BB82D39DC0E3
> curl -sSL https://get.rvm.io | sudo bash -s stable
> sudo usermod -a -G rvm `whoami`
```

La primera comanda instal·la les claus de verificació de paquets. S'utilitza per controlar que el paquet no contingui modificacions indesitjades. Seguidament, s'instal·la RVM en si, i en la tercera s'atribueix els permisos de rvm a l'usuari actual [38].

Un cop RVM està operatiu, s'ha d'instal·lar la versió de ruby corresponent, en el cas del projecte la v2.5.0, i seleccionar-la com a versió per defecte.

```
> rvm install ruby-2.5.0
> rvm --default use ruby-2.5.0
```

12.1.2 Bundler

Per el control de les "*gemes*" s'utilitza el gestor *Bundler*. Per a instal·lar *bundler* només és necessita executar la següent comanda:

```
> gem install bundler --no-rdoc --no-ri
```

12.1.3 NodeJS

L'aplicació utilitza ReactJS, per tant, és necessari tenir NodeJS i npm en el servidor. Per instal·lar-ho, s'ha d'executar les següents comandes:

```
> curl --fail -ssL -o setup-nodejs
https://deb.nodesource.com/setup_8.x &&
> sudo bash setup-nodejs &&
> sudo apt-get install -y nodejs build-essential
```

12.2 Instal·lació del servidor web

12.2.1 Nginx [39]

L'instal·lació d'nginx es fa a través d'*apt* amb la següent comanda:

```
> apt-get install nginx
```

Un cop finalitzat el procés podem comprovar l'estat de la instal·lació. La primera ens permet observar quins permisos s'han atribuït a Nginx en el Firewall de la màquina:

```
> ufw app list
Available applications:
...
  Nginx Full
  Nginx HTTP
  Nginx HTTPS
...
```

- Nginx Full correspon a l'accés al port :80, utilitzat per a connexions HTTP, i al :443 per a connexions HTTPS (TLS/SSL)

- Nginx HTTP només garanteix l'accés al port :80

- Nginx HTTPS només garanteix l'accés al port :443 (TLS/SSL)

I la segona ens permet a comprovar l'estat del servei:

```
> systemctl status nginx

● nginx.service - A high performance web server and a reverse
proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled;
vendor preset: enabled)
   Active: active (running) since ...; ... ago
   ....
```

12.2.2 Phusion Passenger [40]

Primer hem d'instal·lar phusion passenger amb el mòdul d'integració de Nginx:

```
> sudo apt-get install -y dirmngr gnupg
> sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80
--recv-keys 561F9B9CAC40B2F7
> sudo apt-get install -y apt-transport-https ca-certificates
>
> sudo sh -c 'echo deb
https://oss-binaries.phusionpassenger.com/apt/passenger stretch
main > /etc/apt/sources.list.d/passenger.list'
> sudo apt-get update
>
> sudo apt-get install -y libnginx-mod-http-passenger
```

Un cop instal·lat, hem de comprovar que els fitxers de configuració s'han crear corrrectament:

```
> if [ ! -f /etc/nginx/modules-enabled/50-mod-http-passenger.conf
]; then sudo ln -s
/usr/share/nginx/modules-available/mod-http-passenger.load
/etc/nginx/modules-enabled/50-mod-http-passenger.conf ; fi
> sudo ls /etc/nginx/conf.d/mod-http-passenger.conf
```

En cas que no existeixi cap fitxer a:

```
/etc/nginx/conf.d/mod-http-passenger.conf
```

S'ha de crear manualment i afegir-hi les opcions *passenger_ruby* i *passenger_root*:

```
passenger_root
/usr/lib/ruby/vendor_ruby/phusion_passenger/locations.ini;
passenger_ruby /usr/bin/passenger_free_ruby;
```

Un cop realitzades les comprovacions, s'ha de reiniciar Nginx per a que reconeixi el mòdul d'integració.

```
> service nginx restart
```

Per a validar la instal·lació completa i que la integració funcina correctament, s'ha d'executar la següent comanda:

```
> sudo /usr/bin/passenger-config validate-install
 * Checking whether this Phusion Passenger install is in PATH... ✓
 * Checking whether there are no other Phusion Passenger
installations... ✓
```

12.3 Desplegament de l'aplicació [41]

12.3.1 Git

Primer hem de comprovar que tinguem git instal·lat en el servidor, en cas que no hi sigui, s'ha d'instal·lar executant:

```
> apt-get install git
```

12.3.2 "Pull" del codi

Un cop tenim git en el servidor, hem de descarregar el codi. Abans, s'ha de crear la carpeta destí on volem guardar l'aplicació i canviar-ne els permisos:

```
> sudo mkdir -p /var/www/myapp
> sudo chown www-data: /var/www/myapp
```

Un cop tenim la carpeta creada, hem de descarregar el codi del repositori:

```
> cd /var/www/myapp
> sudo -u www-data -H git clone
https://bitbucket.org/username/myapp.git code
```

12.3.4 Instal·lació de dependències

Un cop tenim el codi hem d'instal·lar les dependències especificades en el fitxer de "Gemes" de l'aplicació:

```
> sudo -u myappuser -H bash -l
> rvm use ruby-2.5.0
> bundle install --deployment --without development test
```

12.3.5 Configuració base de dades [42]

Primer cal instal·lar mysql2 en el servidor. Durant el procés d'instal·lació, s'haurà d'introduir la contrasenya de l'usuari "*root*".

```
> sudo apt-get install mysql-server mysql-client
libmysqlclient-dev
```

I, un cop instal·lats tots els paquets, hem d'inicialitzar mysql2 i d'assegurar la instal·lació eliminant valors de configuració que per defecte són vulnerables a atacs:

```
> sudo mysql_install_db
> sudo mysql_secure_installation
```

Finalment, hem de modificar la configuració de connexió RoR amb la base de dades:

```
> nano config/database.yml
```

```
production:
 host: localhost
 adapter: mysql2
 encoding: utf8
 port: 3306
 reconnect: false
 database: db_name
 username: db_username
 password: db_password
 socket: '/var/run/mysqld/mysqld.sock'
```

On db_name és el nom de la base de dades principal i db_user i db_password són l'usuari i contrasenya de connexió a la base de dades. És recomanable crear un usuari diferent de root amb menys permisos per evitar riscos en la seguretat de l'aplicació. Aquest últim pas s'ha de realitzar també per al fitxer */config/database_requests.yml* .

Un cop s'ha configurat, s'ha de generar la base de dades a partir de l'esquema i les migracions, i afegir-hi les dades inicials:

```
> rails db:create
> rails requests:db:create
> rails db:seeds
> rails requests:db:seeds
```

12.3.6 Configuració d'encriptació

Per assegurar la seguretat de les sessions, RoR té una clau amb la que encripta la informació de les sessions d'usuaris actius. La configuració es realitza amb dos pasos.

Primer s'ha de generar una clau secreta mitjançant la comanda:

```
> bundle exec rake secret
```

I després, s'ha d'editar el fitxer *config/secrets.yml* amb la clau generada:

```
> nano config/secrets.yml
```

```
production:
 secret_key_base: my_scret_key
```

12.3.7 Configuració dels WebSockets [43] [44]

L'únic pas per a la configuració és la instal·lació de Redis en el servidor:

```
> sudo apt install redis-server
```

Un cop instal·lat ens hem d'assegurar que, en el fitxer de configuració, la variable "*supervised*" té el valor "*systemd*":

```
> sudo nano /etc/redis/redis.conf
```

```
...
supervised systemd
...
```

Finalment, hem de reiniciar el servei i comprovar-ne l'estat:

```
> sudo systemctl restart redis.service
> sudo systemctl status redis
● redis-server.service - Advanced key-value store
   Loaded: loaded (/lib/systemd/system/redis-server.service;
enabled; vendor preset: enabled)
   Active: active (running) since ...; ... ago
...
```

Un cop comprovat l'estat, és important configurar una contrasenya segura d'accés a les dades. Per a fer-ho s'ha de modificar la variable "*requirepass*" del fitxer "*/etc/redis/redis.conf*" amb una contrasenya forta:

```
> sudo nano /etc/redis/redis.conf
```

```
...
requirepass my_secure_password
...
```

Un cop actualitzat s'ha de reiniciar el servei:

```
> sudo systemctl restart redis.service
```

I es pot comprovar la contrasenya amb les següents comandes:

```
> redis-cli
$
$ set key1 10
(error) NOAUTH Authentication required.
$
$ auth my_secure_password
$ set key1 10
OK
$ exit
>
```

Finalment, s'ha de modificar la configuració de RoR perquè utilitzi Redis amb l'identificació correcte. S'ha de canviar el fitxer "config/cable.yml" en la carpeta de l'aplicació:

```
> nano config/cable.yml
```

```
production:
 adapter: redis
 url: redis://my_secure_password@localhost:6379
```

12.3.8 Configuració del host virtual

Un cop s'han configurat tant la base de dades com dels web sockets, hem de fer que el servidor web reconegui l'aplicació perquè es pugui accedir-hi des de qualsevol navegador. En la carpeta "/etc/nginx/sites-available" s'ha de crear el fitxer de configuració del host virtual:

```
> nano my_app
```

Amb el contingut:

```
server {
    listen 80;
    server_name sub_domain.domain.com;
    root /var/www/my_app/public;
    passenger_enabled on;

    passenger_ruby /usr/local/rvm/gems/ruby-2.5.0/wrappers/ruby;
    passenger_sticky_sessions on;

    location /cable {
        passenger_app_group_name my_cable_name_action_cable;
        passenger_force_max_concurrent_requests_per_process 0;
    }
}
```

Un cop s'ha configurat el host, s'ha d'habilitar mitjançant la comanda:

```
> ln -s /etc/nginx/sites-available/my_app
/etc/nginx/sites-enabled/
```

I reiniciar el servei nginx:

```
> service nginx restart
```

12.3.9 Compilació d'assets [45]

Per acabar, s'han de compilar els assets: fitxers javascript i regles d'estil CSS:

```
> rvm use 2.4.0
> bundle exec rake assets:precompile db:migrate db:seed
RAILS_ENV=production
> bundle exec rake requests:db:migrate RAILS_ENV=production
> sudo chown -R www-data:www-data *
> sudo touch tmp/restart.txt
```

En aquest punt, si es visita el domini s'hauria de poder veure correctament la pàgina d'inici de l'aplicació.

13 Resultats

13.1 Control de gestió

En aquest capítol es detalla com s'han dut a terme els sprints planificats i si s'han complert els objectius temporals i tasques realitzades. Per tal de realitzar el control de gestió s'ha realitzat un seguiment de les hores les quals s'han treballat. Per a cada sprint es mostra les hores treballades segon el previst en els diagrames de GANTT i s'ha calculat quin és el valor dels índexs destacats en l'explicació del control de gestió del punt 7.3.3.

Sprint 1

Setmana	Hores realitzades
20.08.18 - 26.08.18	16:30
27.08.18 - 02.09.18	20:30
03.09.18 - 09.09.18	20:00
Total	57:00 (de 57:00 planificades)
Index	**Valor**
Càrrega de treball	1.00
Productivitat	1.00
Costos directes	1.00
Costos indirectes	1.00

Taula 34. Resum sprint 1

Els quatre indicadors estan en el seu valor òptim. Per tant podem concloure que en aquest sprint s'han gestionat de forma correcta els recursos materials i s'ha realitzat una bona distribució de tasques amb un pes adequat a cada una.

Sprint 2

Setmana	Hores realitzades
10.09.18 - 16.09.18	19:00
17.09.18 - 23.09.18	20:30
24.09.18 - 30.09.18	20:00
Total	59:30 (de 57:00 planificades)
Index	**Valor**
Càrrega de treball	1.04
Productivitat	1.00
Costos directes	1.04
Costos indirectes	1.03

Taula 35. Resum sprint 1

En l'sprint 2 es van realitzar més hores de les previstes. Tot i això els indicadors es troben dins del marge permès del 10%. L'sprint es va dur a terme amb normalitat.

Sprint 3

Setmana	Hores realitzades
01.10.18 - 07.10.18	20:00
08.10.18 - 14.10.18	20:30
15.10.18 - 21.10.18	16:30
Total	57:00 (de 57:00 planificades)
Index	**Valor**
Càrrega de treball	1.00
Productivitat	1.00
Costos directes	1.00
Costos indirectes	0.98

Taula 36. Resum sprint 3

L'sprint 3 es va ajustar a la planificació. Es van realitzar les tasques planificades i es van dedicar només quatre minuts més, respecte de la planificació, per a dur-les a terme.

Sprint 4

Setmana	Hores realitzades
22.10.18 - 28.10.18	20:00
29.10.18 - 04.11.18	20:00
05.11.18 - 11.11.18	20:00
Total	60:00 (de 57 planificades)
Index	**Valor**
Càrrega de treball	1.04
Productivitat	1.00
Costos directes	1.04
Costos indirectes	1.03

Taula 37. Resum sprint 4

Per a realitzar l'sprint 4 es van necessitar més hores de les planificades. Tot i això es va mantenir per sota del marge permès del 10%. Aquest augment en el nombre d'hores treballades va ser conseqüència del grau de dificultat que va suposar fer la re-estructuració de les dades i adaptar l'script de normalització perquè continues funcionant sense errors.

Sprint 5

Setmana	Hores realitzades
12.11.18 - 18.11.18	20:00
19.11.18 - 25.11.18	19:00
26.11.18 - 02.12.18	21:00
Total	60:00 (de 57 planificades)
Index	**Valor**
Càrrega de treball	1.04
Productivitat	1.00
Costos directes	1.04
Costos indirectes	1.03

Taula 38. Resum sprint 5

En l'sprint 5 també es van superar el nombre d'hores treballades. Amb el temps dedicat es van poder dur a terme les tasques necessàries.

Sprint 6

Setmana	Hores realitzades
03.12.18 - 09.12.18	19:30
10.12.18 - 16.12.18	19:30
17.12.18 - 23.12.18	19:00
Total	58:00 (de 57 planificades)
Index	**Valor**
Càrrega de treball	1.02
Productivitat	1.00
Costos directes	1.02
Costos indirectes	1.01

Taula 39. Resum sprint 6

Durant l'sprint 6 es van completar totes les tasques planificades. Tot i això, va ser necessari invertir-hi poc més d'una hora addicional respecte el previst.

Sprint 7

Setmana	Hores realitzades
24.12.18 - 30.12.18	12:00
31.12.18 - 06.01.19	23:00
07.12.19 - 13.12.19	23:00
Total	58:00 (de 58 planificades)
Index	**Valor**
Càrrega de treball	1.00
Productivitat	1.00
Costos directes	1.00
Costos indirectes	1.00

Taula 40. Resum sprint 7

En el 7è i últim sprint es va complir amb la planificació de forma exacta. Es va treballar el temps suficient i necessari per a fer la totalitat de les tasques previstes.

Resum global

	Hores
Realitzades	409:00
Planificades	400
Index	**Valor**
Càrrega de treball	1.02
Productivitat	1.00
Costos directes	1.02
Costos indirectes	1.01

Taula 41. Resum sprints

En el global del projecte s'hi han dedicat més hores de les que s'havien previst en un inici en la planificació temporal d'aquest. Tot i això, aquest increment ha sigut mínim (d'un 2%) i es troba dins del marge de contingència marcat al 10%. Per aquest motiu podem concloure que el treball s'ha adaptat al previst i no ha implicat cap sobrecost no hagués estat contemplat en el pressupost inicial.

13.2 Sostenibilitat

13.2.1 Autoavaluació de la competència

En un projecte en l'entorn de les TIC és important tenir en consideració els tres aspectes que afecten la sostenibilitat d'aquests. I a més amb un èmfasi especial degut al propi entorn del sector, el qual ha estat sempre pioner en desenvolupar avenços.

En aquest sentit és normal que l'interès personal en oferir solucions encarades a millorar, d'una forma o altra, la societat, ja sigui amb aplicacions que aportin un gran valor nou o amb d'altres que senzillament, contribueixin a millorar eines ja existents. L'exemple principal és la cooperació que s'està fent en aquest TFG: la reforma d'una part crucial per a un projecte com és GreenCustomers.

Tot i la col·laboració ja realitzada, i l'actual, és encara difícil poder preveure de forma suficientment clara quines són les afectacions que tindrà el desenvolupament d'un projecte. I més concretament sobre la dimensió ambiental, ja que és la que té menys dependències directes en comparació en les altres dues dimensions.

Pel que fa a la dimensió econòmica, des de el meu punt de vista, és la més òbvia. El senzill fet de convertir una idea en un producte és indispensable que aquest no esdevingui un forat sense fons en terme de pressupost. No és necessari desenvolupar un nou article si es pot aconseguir un mateix resultat utilitzant una eina ja existent o la suma d'eines amb funcions específiques i compatibles.

La dimensió social, per contrast amb l'econòmica que es té en compte principalment en l'inici del projecte, s'ha de tenir en compte durant el transcurs d'aquest. És important la satisfacció i el benestar dels actors que hi estan implicats. No és bona idea una sobrecàrrega ni un entorn de treball tens.

I per últim la dimensió ambiental. Aquest es deslliura del projecte en si, ja que es veu afectada amb un grau alt dels recursos que s'usen durant el projecte. És difícil poder predir afectacions sobre elements fora de l'abast d'un mateix. No es pot avaluar quin impacte té, per exemple, el servidor contractat si no es pot saber qui, quan ni com es va fabricar i quin és el seu consum de recursos.

Per tant, tot i poder tenir intuïcions i nocions de com pot afectar en major o menor grau la presa de decisions sobre la sostenibilitat, encara m'és necessària experiència i coneixements per acotar millor les prediccions.

13.2.2 Dimensió Econòmica

Aquest projecte es basa en la millora d'una infraestructura existent per a dos motius bàsics: la preparació per a possibles nous requeriments que es puguin implementar més fàcilment degut al compliment dels estàndard de canviabilitat i adaptabilitat del software, però per altra banda per a l'automatització de processos que actualment es realitzen manualment i que per tant requereixen un cost de recursos humans.

Per tant, tot i la despesa inicial que representa, aquest està pensat per a poder reduir els costos directes futurs. L'efecte esperat és poder amortitzar el projecte amb l'estalvi que generi i els nous ingressos directes de la posada en marxa en producció.

13.2.3 Dimensió Ambiental

Per els propis motius en què es mou el projecte, aquest no invertirà en eines físiques sinó que reutilitzaran les eines externes. El principal cas és el servidor. L'empresa ja té contractats un servidor que és capaç de mantenir la nova infraestructura amb capacitat suficient de creixement.

La resta d'eines usades són *software*, en conseqüència, el servidor es converteix en l'únic autor de la petjada mediambiental sobre el món físic. I aquest només es pot reduir si s'augmenta l'eficiència del *hardware* que el compon.

13.2.4 Dimensió Social

En aquest cas, la dimensió social, se centra en la satisfacció per part de l'equip de GreenCustomers. L'aplicació resultant del projecte és un punt d'ancoratge més fort per avançar i fer créixer l'empresa. Significa tenir una eina completament personalitzable i a mida, que s'adapta a les necessitats de l'empresa i a les noves que puguin sorgir en el transcurs de la seva vida útil. És per aquests motius que la satisfacció de l'equip es veu molt afectada positivament respecte a l'expressada amb el sistema antic.

Personalment, és un objectiu important i una batalla guanyada contra mi mateix. El fet de comptar amb una eina ben planificada des de l'inici serveix per poder esmenar errors o decisions equivocades degut a la falta de coneixement durant els primers mesos en què vaig participar en l'empresa. Amb el coneixement adquirit en aquest temps, així com amb l'experiència de treballar en un entorn dinàmic com és el de les Start-Ups no només m'ha aportat experiència en el món laboral, sinó que addicionalment he pogut conviure en un entorn diferent de l'empresa tradicional.

13.2.5 Matriu de sostenibilitat

La matriu de sostenibilitat en projectes d'enginyeria és una eina que resumeix qualitativament els aspectes mediambientals que aquest té en compte.

La matriu es compon d'una taula de tres columnes on es tenen en compte la posada en producció (PPP) del projecte, la vida útil d'aquest, i els riscos que el poden afectar. I l'altre eix és cada una de les tres dimensions esmentades en els apartats anteriors. [46]

La matriu té la següent estructura amb el rang de valors possibles:

	PPP	Vida útil	Riscos
D. Econòmica	0:10	0:20	-20:0
D. Ambiental	0:10	0:20	-20:0
D. Social	0:10	0:20	-20:0
Sostenibilitat	0:30	0:60	-60:0
	-60:90		

Taula 42. Estructura matriu de sostenibilitat

Tenint en compte l'argumentació dels efectes de cada dimensió que preveu le projecte, s'han atorgat el següents punts en cada cassella:

	PPP	Vida útil	Riscos
D. Econòmica	6	14	-9
D. Ambiental	6	12	-7
D. Social	9	17	-4
Sostenibilitat	21	43	-20
	44 (-60:90)		

Taula 45. Matriu de sostenibilitat

14 Justificació del projecte

14.1 Objectius i Abast

El projecte que s'ha desenvolupat és tota una plataforma amb tres punts claus per la renovació del core de la infraestructura de la StartUp GreenCustomers.

El primer punt és una aplicació web per a la gestió del producte que ofereix l'empresa: les *greenrooms*. Aquesta aplicació permet controlar de forma fàcil els components i elements que configuren la *greenroom*. Cada un d'ells compte amb el seu propi panell de gestió on, a partir de formularis, es poden realitzar les funcions bàsiques CRUD: crear elements nous, poder-ne llegir-ne el contingut, actualitzar-lo, i en última instància eliminar-lo del sistema.

Addicionalment, alguns d'aquests panells permeten funcions específiques de cada model per a facilitar i agilitar l'experiència d'usuari. Com és el cas descrit en la capacitat de jerarquitzar les zones.

El segon se centra en el desenvolupament del *end-point* de la tecnologia IoT implicada. L'entrada de totes les dades generades pels sensors de les greenrooms al sistema. Aquest punt s'encarrega del tractament de les dades: convertir-les en informació amb valor real amb les quals se'n pugui extreure coneixement.

Aquest inclou també tot l'emmagatzematge de les dades i el control d'errors i gestió de les alarmes generades. A més de la creació d'un registre que permet la recuperació ràpida i eficaç de les trames en cas de pèrdua.

El tercer i últim punt és el desenvolupament d'una API REST encarada a explotar la data procedent del sistema IoT. Aquesta API és l'únic punt de contacte amb la resta de la infraestructura del GreenCustomers.

A part dels propis *end-point* de la interfície, s'ha programat la gestió de permisos per a poder accedir-hi de forma segura mantenint la privacitat de les dades de cada client.

Aquest treball es va pensar perquè tots els objectius marcats es puguin realitzar dins del temps acotat i per tant, l'abast és el que es va definir en els objectius. El projecte, un cop posat en producció, serà la primera versió de les que puguin esdevenir. No és una infraestructura estàtica. L'entorn dinàmic que s'ha definit durant la documentació presenta de forma clara i assegura que nous requisits i funcionalitats s'hauran d'afegir en un futur sense cap termini especificat.

14.2 Coneixements previs

Durant l'especialitat de software he acumulat coneixements que s'estan posant en pràctica en aquest projecte.

Primerament, l'assignatura troncal de l'especialitat com és **Arquitectura del Software** (AS), més la predecessora Introducció a l'**Enginyeria del Software** (IES), són claus per a poder dissenyar tot un sistema que compleixi, com s'ha dit en el llarg del plantejament inicial del projecte, amb els estàndards. Un software canviable, mantenible. Que sigui capaç d'adaptar-se a noves necessitats.

L'assignatura de **Gestió de Projectes del Software** (GPS), va ser imprescindible per a conèixer els diferents mètodes de treball àgils i saber com posar-los en pràctica. En aquest treball, el mètode Scrum.

En un projecte com aquests, ha sigut necessari cursar **Enginyeria de Requisits** (ER). Al tractar-se de la renovació d'una infraestructura existent, és bàsic tenir els coneixements de com saber quines són les peces clau del nou software: Que és el que no funciona actualment i s'ha de reparar, què és el que no és necessari canviar, i que és el que s'ha d'incorporar.

L'assignatura d'**Arquitectura de Serveis Web** (ASW) ha permès saber com s'ha de documentar i posteriorment desenvolupar una API de tipus REST. Després d'haver pogut treballar de forma teòrica i pràctica en l'assignatura, construir una API decent per al projecte no presenta una dificultat.

Disseny de Bases de Dades (DBD), m'ha permès tenir en compte com poder explotar més eficientment una base de dades relacional. A més, el fet d'haver sigut estudiant de **Conceptes de Bases de Dades Especialitzades** (CBDE), m'ha aportat saber triar quin és el tipus de base de dades més adient per a cada necessitat, i a desconfiar de les meravelles de les NoSQL si ens és desconeguda.

A **Conceptes Avançats de Programació** (CAP), se'ns va ensenyar formes per a realitzar una programació més eficient. Profunditzant més no en com fer algoritmes amb cost computacionalment baixos, sinó en saber explotar el llenguatge triat.

I per últim, en l'assignatura de **Projecte d'Enginyeria del Software** (PES), em va servir com a reforçament per a posar en pràctica totes les experiències i coneixements obtinguts en les assignatures esmentades, i com a prèvia del projecte del TFG.

14.3 Especialitat

Amb totes les assignatures implicades, els objectius del projecte i les fases clares d'especificació, disseny i implementació, deixen clar que és un treball que abasta, en major o menor mesura tots els àmbits de l'especialitat.

Des de saber gestionar la planificació i especificació d'un projecte com a enginyer de requisits, analista o gestor de projectes, a com gestionar correctament bases de dades i dissenyar un bon sistema com a arquitecte del software.

Posar en pràctica tots els rols possibles sorgits de l'especialitat d'Enginyeria del Software en un cas com és el plantejat en el projecte, crec que és la millor forma de sintetitzar els coneixements adquirits durant els últims dos anys. I viceversa, que aquest projecte s'adapta, correctament als resultats esperats després d'haver cursat l'especialitat.

14.4 Justificació de competències

- **CES1.1** *(Desenvolupar, mantenir i avaluar sistemes i serveis software complexos i/o crítics.)* - *(Alt)* El projecte se centra en la renovació del que és considerat el *core* de GreenCustomers. L'eina essencial per a la pròpia existència del que defineix l'empresa. Per tant és un software que no ha de fallar, i en cas que hi aparegués un error, aquest hauria de ser ràpidament detectat i solucionat per a garantir el correcte funcionament.

- **CES1.2** *(Donar solució a problemes d'integració en funció de les estratègies, dels estàndards i de les tecnologies disponibles.)* - *(Mitjana)* El *backoffice* que es dissenyarà haurà de comunicar-se i permetre la comunicació amb altres sistemes. Aquesta comunicació vindrà marcada per protocols establerts com és el del proveïdor de la xarxa IoT, o per connexions entre els diferents components de la infraestructura interna.

- **CES1.5** *(Especificar, dissenyar, implementar i avaluar bases de dades.) - (Baix)* Diferents recursos es reutilitzaran o se li aplicaran canvis menors. Un dels casos principals és el disseny de la base de dades. Tot i tenir una part important com és l'emmagatzematge de la *data* procedent del IoT, la complexitat és quasi inexistent.

- **CES1.7** *(Controlar la qualitat i dissenyar proves en la producció de software.) - (Alt)* Aquest software, a diferència del que s'ha reemplaçat, compte amb un pla de proves com a garantia que el software es manté estructurat i clar. Dues característiques de què aquest és fàcilment mantenible i comprensible.

- **CES2.1:** *(Definir i gestionar els requisits d'un sistema software.) - (Mitjana)* De la mateixa manera que amb la que s'ha reutilitza part de l'estructura de la base de dades, molts dels requisits del sistema, principalment funcionals, no s'han modificat. Tot i això es destaca la importància que va significar detectar què és el que no funcionava a nivell d'usuari: què és el que el sistema actual no permetia o, sí que ho feia, però que no complia amb totes les necessitats.

15 Conclusions

15.1 Futures versions

En futures versions de l'aplicació es contempla en tres fases. La primera és la transformació completa del portal per als hotelers. Transformar l'actual que es compon d'un back-end i un front-end en una Single Page Application (SPA), on només existeixi un front-end connectat amb l'API desenvolupada en aquest projecte i que permet obtenir els mateixos resultats que el sistema actual. El nou portal es convertiria en un panell de control d'estil dashboard. D'aquesta manera tant l'arquitectura com la mantenibilitat del portal se simplificarien de manera substancial.

La segona fase és la creació d'un sistema de notificacions d'alertes en temps real. Una alerta, a diferència de les alarmes, són avisos personalitzables per a cada hotel i que responen a nivells de consums majors a un llindar durant un període de temps. Aquestes notificacions es realitzarien per correu electrònic tant als encarregats dels hotels com a l'equip de GreenCustomers, i per notificacions *in-app* en el backoffice i el portal.

La tercera fase consistiria en reformar l'aplicació web orientada als hostes. Les modificacions simplificarien l'estructura de dades i components per adaptar-ho al nou backoffice. A més incorporaria noves funcionalitats per a convertir l'actual web-app en una Progressive Web App (PWA):

- Aplicació en *home-screen*: Simular el comportament d'una aplicació nativa creant un accés directe a les pàgines d'aplicacions dels dispositius mòbils.

- *Push-notifiacions*: Utilitzar el sistema natiu de notificacions per avisar l'hoste d'esdeveniments o recordatoris de l'app.

- Mode offline: Permetre a l'usuari revisar les últimes dades consultades i altres funcionalitats bàsiques en mode offline.

15.2 Valoració personal

Aquest projecte és per a mi un repte complet. El primer sistema desenvolupat de GreenCustomers va créixer amb decisions errònies per falta de coneixement i experiència. Poder reparar aquests errors no era trivial, ja que modificaven tota l'estructura de dades, part de l'estructura de components i l'arquitectura del sistema i no es podia realitzar amb canvis menors introduïts de mica en mica, s'havia d'afrontar global i directament. Ha sigut un projecte ambiciós, no només per la complexitat que comportava, sinó per tenir un marge de maniobra curt degut al calendari que limitava el temps de desenvolupament a poc més de 5 mesos.

També ha sigut una font d'autoinspiració i motivació per a seguir treballant amb Green Customers i seguir guanyant coneixement i experiència. En aquest projecte hi he bolcat tot el que he après en els últims tres anys per a evitar caure en els mateixos errors, però també he après l'ús de noves eines com són les notificacions en temps real.

La realització del projecte, per tant, ha sigut fer un pas endavant i marcar l'inici d'una nova etapa: passar d'una etapa on vaig començar des de gairebé re, a una on puc aplicar el que sé i on em veig capaç d'emprendre nous projectes i nous reptes amb més independència i amb més eines.

Apèndix 1. API vs WebHook

Una API, o Application Programming Interface en les seves sigles en anglès, és un conjunt de funcions i/o protocols que permet la comunicació entre blocs de diferents softwares [47]. En altres paraules, és un conjunt de mètodes pertanyents a una aplicació i que són, en un principi, accessibles a tothom.

La mecànica és senzilla, un programa extern realitza una petició al software en qüestió i aquest respon a l'acció requerida.

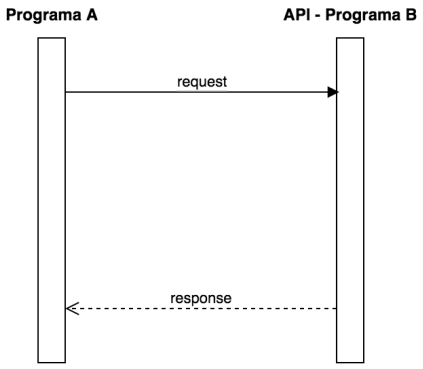


Diagrama del funcionament d'una API

Un WebHook és un mètode que augmenta o canvia el comportament d'una petició a una aplicació web. Com pot ser peticions a una API [48]. En aquest escenari, poden haver-hi dos o tres actors implicats. Pel cas d'aquest treball en són tres.

Tenim un software A que realitza una petició a un software B a través de la seva API. El programa B, gestiona la petició en envia la resposta al programa A. Paral·lelament, de forma synchrona o asynchorna, realitza una petició a un software C com a notificació de la rebuda d'una petició.

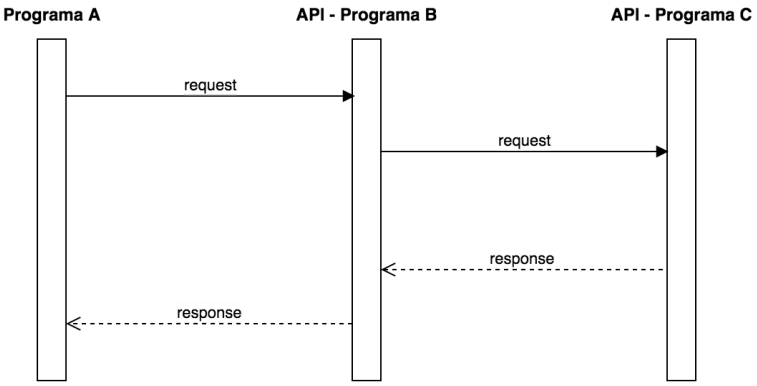
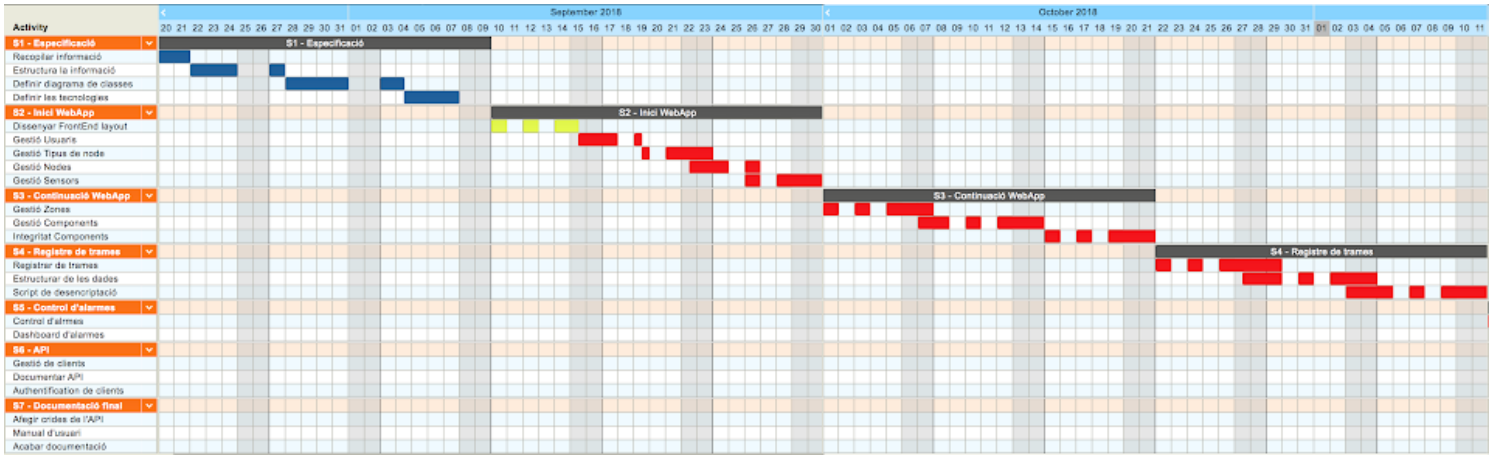
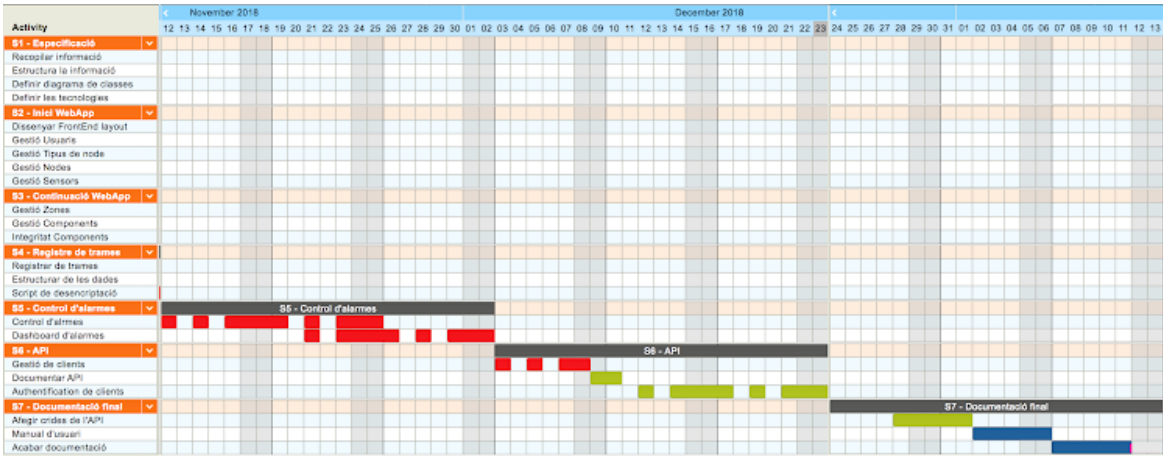


Diagrama del funcionament d'un WebHook

Apèndix 2. Diagrama de GANTT





Apèndix 3. Estimació costos directes

Sprint	Activitat	Import			
		Hores Totals Activitat	h. CEO / CIO	h. Dev.	Import (€)
1	Recopilar Informació	8	8	8	224
	Estructura la informació	15	15	15	420
	Definir diagrama de classes	21	21	21	588
	Definir les tecnologies	13	0	13	104
	Total sprint 1	57	44	57	1336
2	Dissenyar FrontEnd layout	13	0	13	104
	Gestió Usuaris	12	0	12	96
	Gestió Tipus de node	10	0	10	80
	Gestió Nodes	9	0	9	72
	Gestió Sensors	13	0	13	104
	Total sprint 2	57	0	57	456
3	Gestió Zones	17	0	17	136
	Gestió Components	21	0	21	168
	Integritat Components	19	0	19	152
	Total sprint 3	57	0	57	456
4	Registrar de trames	18	0	18	144
	Estructurar de les dades	16	0	16	128
	Script de desencriptació	23	0	23	184
	Total sprint 4	57	0	57	456
5	Control d'alarmes	30,5	2	30,5	284
	Dashboard d'alarmes	26,5	2	26,5	252
	Total sprint 5	57	4	57	536
6	Gestió de clients	15	0	15	120
	Documentar API	8	8	8	224
	Authentification de clients	34	0	34	272
	Total sprint 6	57	8	57	616
7	Afegir crides de l'API	19	0	19	152
	Manual d'usuari	19	8	19	312
	Acabar documentació	20	0	20	160
	Total sprint 7	58	8	58	624
	Total	400	64	400	4480

Referències

[1] <<Our Vision | Sigfox>> Sigfox, Inc. 8 de Setembre. [En Linea]. Disponible: https://www.sigfox.com/en/our-vision

[2] <<About LoRa Alliance>> LoRa Alliance ™, 8 de Setembre. [En Linea]. Disponible: https://lora-alliance.org/about-lora-alliance

[3] <<About LoRaWAN>> LoRaWAN ™, 8 de Setembre. [En Linea]. Disponible: https://lora-alliance.org/about-lorawan

[4] <<CodeIgniter>> British Columbia Institute of Technology, 18 de Novembre. [En Linea]. Dispible: https://codeigniter.com/

[5] <<ISO/IEC 25010:2011>> International Organization for Standardization, 4 de Febrer. [En Linea] Disponible: https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en

[6] <<Ruby On Rails>> Ruby on Rails, 13 de Febrer. [En Linea] Disponible: https://rubyonrails.org/

[7] <<Devise>> GitHub - plataformatec/devise, 13 de Febrer. [En Linea] Disponible: https://github.com/plataformatec/devise

[8] <<React Rails>> Github - reactjs/react-railss, 13 de Febrer. [En Linea] Dispoinible: https://github.com/reactjs/react-rails

[9] <<ReactJS>> Facebook, Inc., 13 de Febrer. [En Linea] Disponible: https://reactjs.org/

[10] <<Lean StartUp>> Wikipedia Foundation, Inc., 28 d'Agost. [En Linea]. Disponible: https://es.wikipedia.org/wiki/Lean_startup

[11] <<Agile Software Development>> Wikipedia Foundation, Inc., 13 de Febrer. [En Linea] Disponible: https://en.wikipedia.org/wiki/Agile_software_development

[12] <<AgileManifesto>> Jim Highsmith - Manifesto for the Agile Software Development, 13 de Febrer. [En Linea] Disponible: http://agilemanifesto.org/

[13] <<Visual Code Studio>> Microsoft Corporation, Inc. 29 de Setembre. [En Linea]. Disponible: https://code.visualstudio.com/

[14] <<Git>> *Open Source Software*. 29 de Setembre. [En Linea]. Disponible: https://git-scm.com/

[15] <<Docker>> Docker, Inc. 29 de Setembre. [En Linea]. Disponible: https://www.docker.com/

[16] <<Postman>> Postdot Technologies Pvt. Ltd. 29 de Setembre. [En Linea]. Disponible: https://www.getpostman.com/

[17] <<Google Docs>> Google LLC. 29 de Setembre. [En Linea]. Disponible: https://www.google.com/intl/en_EN/docs/about/

[18] <<Tom's planner>> Toms Planner N.V. 29 de Setembre. [En Linea]. Disponible: https://www.tomsplanner.com/

[19] <<Cost Indirecte>> Wikipedia Foundation, Inc., 5 d'Octubre . [En Linea]. Disponible: https://ca.wikipedia.org/wiki/Cost_indirecte

[20] <<Real Decreto 1777/2004, de 30 de julio>> *Boletín Oficial del Estado*. 5 d'Octubre. [En Linea]. Disponible: https://boe.es/buscar/act.php?id=BOE-A-2004-14600&p=20141128&tn=2

[21] <<Single page Application>> pshrnm blog, 13 de Febrer. [En Linea] Disponible: https://blog.pshrmn.com/entry/how-single-page-applications-work/

[22] <<Docker Container>> Docker, Inc., 7 de Febrer. [En Linea] Disponible: https://www.docker.com/resources/what-container

[23] <<Docker (software)>> Wikipedia Foundation, Inc., 7 de Febrer. [En Linea] Disponible: https://en.wikipedia.org/wiki/Docker_(software)

[24] <<Debian Project>> Software in the Public Interest, Inc., 9 de Febrer. [En Linea] Disponible: https://www.debian.org/intro/about#what

[25] <<SQL>> Wikipedia Foundation, Inc., 13 de Febrer. [En Linea] Disponible: https://en.wikipedia.org/wiki/SQL

[26] <<REST - MDN Web Docs>> Mozilla Corporation, 11 de Febrer. [En Linea] Disponible: https://developer.mozilla.org/en-US/docs/Glossary/REST

[27] <<JSON>> Javascript Object Notation - ECMAScript 404, 11 de Febrer [En Linea] Disponible: https://json.org/

[28] <<Simple Mail Transfer Protocol>> Wikipedia Foundation, Inc., 13 de Febrer. [En Linea] Disponible: https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

[29] <<Action Mailer - Ruby On Rails>> Ruby on Rails, 13 de Febrer. [En Linea] Disponible: https://guides.rubyonrails.org/action_mailer_basics.html

[30] <<Active Record Pattern>> Wikipedia Foundation, Inc., 11 de Febrer. [En Linea] Disponible: https://en.wikipedia.org/wiki/Active_record_pattern

[31] <<Active Record - Ruby On Rails>> Ruby on Rails, 13 de Febrer. [En Linea] Disponible: https://guides.rubyonrails.org/active_record_basics.html#the-active-record-pattern

[32] <<Action Controller- Ruby On Rails>> Ruby on Rails, 13 de Febrer. [En Linea] Disponible: https://guides.rubyonrails.org/action_controller_overview.html

[33] <<Action View Ruby On Rails>> Ruby on Rails, 13 de Febrer. [En Linea] Disponible: https://guides.rubyonrails.org/action_view_overview.html

[34] <<Web Sockets>> Wikipedia Foundation, Inc., 13 de Febrer. [En Linea] Disponible: https://en.wikipedia.org/wiki/WebSocket

[35] <<Web Sockets - MDN Web Docs>> Mozilla Corporation, 14 de Febrer. [En Linea] Disponible: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

[36] <<Redis>> Redis Labs, 11 de Feber. [En Linea] Disponible: https://redis.io/

[37] <<Installing Ruby with RVM>> Passenger Registered, 17 de Febrer. [En Linea] Disponible: https://www.phusionpassenger.com/library/walkthroughs/deploy/ruby/ownserver/nginx/oss/install_language_runtime.html

[38] <<RVM: Ruby Version Manager>> © under Michal Papis (2011-2018), Piotr Kuczynski. [En Linea] Disponible: https://rvm.io/rvm/install

[39] <<How To Install Nginx on Debian 9>> DigitalOcean, Inc. 17 de Febrer. [En Linea] Disponible: https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-debian-9

[40] <<Install Passenger + Nginx>> Passenger Registered, 17 de Febrer. [En Linea] Disponible: https://www.phusionpassenger.com/library/walkthroughs/deploy/ruby/ownserver/nginx/oss/stretch/install_passenger.html

[41] <<Deploying a Ruby app on Debian>> Passenger Registered, 17 de Febrer. [En Linea] Disponible: https://www.phusionpassenger.com/library/walkthroughs/deploy/ruby/ownserver/nginx/oss/stretch/deploy_app.html

[42] <<How To Use MySQL with your Ruby On Rails Application>> DigitalOcean, Inc. 17 de Febrer. [En Linea] Disponible: https://www.digitalocean.com/community/tutorials/how-to-use-mysql-with-your-ruby-on-rails-application-on-ubuntu-14-04

[43] <<Integration Rails Action Cable with Passenger + Nginx>> Passenger Registered, 17 de Febrer. [En Linea] Disponible: https://www.phusionpassenger.com/library/config/nginx/action_cable_integration/

[44] <<How To install and secure redis>> DigitalOcean, Inc. 17 de Febrer. [En Linea] Disponible: https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-redis-on-ubuntu-18-04

[45] <<Deploy Applications Updates>> Passenger Registered, 17 de Febrer. [En Linea] Disponible: https://www.phusionpassenger.com/library/walkthroughs/deploy/ruby/ownserver/nginx/oss/deploy_updates.html

[46] <<La sostenibilidad en los proyectos de ingeniería>> Jordi Garcia, Helena García, David López, Fermín Sánchez, Eva Vidal, Marc Alier y Jose Cabré, 7 d'Octubre. [En Linea]. Disponible: http://aenui.net/ojs/index.php?journal=revision&page=article&op=viewArticle&path%5B%5D=127&path%5B%5D=212

[47] <<WebHook> Wikipedia Foundation, Inc., 19 de Novembre. [En Linea]. Disponible: https://en.wikipedia.org/wiki/Application_programming_interface

[48] <<WebHook> Wikipedia Foundation, Inc., 19 de Novembre. [En Linea]. Disponible: https://en.wikipedia.org/wiki/Webhook